2010年度　修士論文

発熱量の低減を目的としたフロアプラニング手法に関する研究

指導教員　大附 辰夫 教授

早稲田大学大学院　基幹理工学研究科
情報理工学専攻

5109B101–5

渡辺 哲也

2011年2月4日

目 次

第1章

序論

1.1 本論文の背景と意義

プロセッサの性能や集積度は日々向上している．クロック周波数は，昔は MHz で限界だったものが今では GHz が当たり前となっており，また，チップサイズも微細化技術の進歩によりどんどん小型化している．一方で，これらの動きに伴い問題となってくるのが，プロセッサの消費電力密度の増大と，それによって引き起こされるチップの発熱問題である．チップが過度の発熱を起こすと，冷却コストやリーク電力などが増え，それにより性能や信頼性，そして寿命は低下してしまう [18]．つまり，プロセッサの性能や集積度の向上がこのままの推移で続く限り，発熱がチップ設計における最大のボトルネックになってしまう事は明らかである．

上記の理由からプロセッサにおける熱制御手法が求められているが，既存研究はその大半が DVFS（Dynamic Voltage and Frequency Scaling）によるものであり，これらの手法はチップ温度が閾値を超えた時に周波数や電圧を下げて温度の低減を図る手法であるため，根本的な解決には至っていない [6][13][19][20]．

これらの理由から，本研究では，チップ内モジュールの配置によって，チップ全体の発熱度合いが大きく異なる事に着目し，発熱を考慮したフロアプラニング手法について考える．

従来手法について説明する．既存のフロアプラニングの方法には大きく分けて SA 法と GA 法の 2 種類がある．SA とはシミュレーテッドアニーリング（Simulated Annealing）の略で，多目的最適化問題を解くための確率的アルゴリズムである [8]．SA 法では任意の初期解から始めて，隣接解の生成・探索を繰り返す事により最適解の発見を試みる．SA 法を用いた熱考慮フロアプラニングの手法には文献 [9][10][11] などがある．これらの手法では，コスト関数で面積，配線，温度の 3 要素を考慮し，その関数に沿って SA 法を実行している．温度の計算方法は文献によって異なるが，出てきた解のコストを計算し，許容の可否を決めるという点では同じである．チップの温度はホットモジュール周りの局所的な配置に依存するため，それを考慮していないという点でこれらの手法は十分とは言えない．

GA とは遺伝的アルゴリズム（Genetic Algorithm）の略で，SA と同じく多目的最適化問題を解くためのアルゴリズムである [14]．GA 法は生物の進化を模して作られたアルゴリズムであり，コンピュータ上で進化の過程（交叉・突然変異・選択）をなぞる様にシミュレーションを行っていく事で最適解に近づくアルゴリズムである．GA 法を用いたフロアプラニングの手法には文献 [1][5][7][15][16] などがある．また，熱を考慮した GA のフロアプラニングの手法には文献 [2][3][12] などがあるが，これらの手法も従来の GA による面積，配線最適化手法に温度という最適化パラメータを追加しただけのものであるため，SA での従来手法と同じく不十分である．

しかし，GA は両親から子となる個体を生成するため，自分自身を変化させていく SA と

比べて，相手親の優れた部分を引き継ぐ事が出来るし，局所最適に陥りにくいというメリットがある．更に，[16] で提案されているエリート度に基づく遺伝的アルゴリズムは，優れた局所配置を受け継ぐ個体をエリートと定義し，エリート個体をなるべく壊さないような遺伝方法を用いているため，この手法自体は熱を考慮していないが，ホットモジュール周りの局所配置に重きを置く熱考慮フロアプラニングに適していると考えられる．

そこで本研究では，第 2 章で詳しく説明するが，このエリート度に基づく遺伝的アルゴリズムを用い，そこに新たな評価関数の設定と局所改良法を導入する事でより低温なフロアプランを目指す．また，GA により求めたフロアプランに対して，更にホットモジュール周りに着目してモジュールの置き換えを行う事で更なる低温化を図る．

本研究の目的である発熱量の低減とは，チップ内で最も高温となるモジュールの動作温度を低減させる事を意味する．つまり，チップの最高温度を下げる事が本研究の目的である．提案する手法により本研究の目的が達成されれば，最も低温での動作が可能なチップのフロアプランを求める事が出来る．

1.2 本論文の概要

本論文では最も低温で動作出来るチップのフロアプランを求めるアルゴリズムとして，エリート度に基づく遺伝的アルゴリズムにおける新たな評価関数の設定及び局所改良法と，GA 終了後の配置に対するモジュール置き換え手法を提案する．本論文は 5 章から構成され，以下に各章の概要を紹介する．

第 2 章「**エリート度に基づく遺伝的アルゴリズム**」では，本研究のベースとなる手法であるエリート度に基づく遺伝的アルゴリズムについて，その特徴を説明する．ここで言うエリートとは，ある世代において一定値以上の評価値を持つ個体の事である．また，エリート度とは，先祖にどれだけの数のエリート個体を持つかを基準に計算される．

第 2 章では、まず初めにこの手法で用いるデータ構造表現であるシーケンスペアについて述べ，次に，評価値の計算及び局所改良，スケーリングとルーレット選択，そしてエリートの定義とエリート度の計算方法について説明する．更に，エリート度に基づく交叉について説明を行う．交叉方法は，エリート交叉（CPTX）と非エリート交叉（PPEX）の 2 種類がある．突然変異と局所改良についても説明を行う．

第 3 章「**提案フロアプラニング手法**」では，動作温度の計算方法と評価関数の設定方法，局所改良法について説明する．動作温度は，いくつかのモジュールを監視対象とし，そのモジュール郡の熱移動の合計値を消費電力密度による近似で求める．評価関数に関しては，動作温度にフォーカスしたいので，その他で余分なものについては出来るだけ考えないようにしたい．そのため，面積は評価関数から除外し，制約として与える事とする．また，モジュール置き換え手法の内容についても説明する．モジュール置き換えは，GA によって出力されたフロアプランに対して行い，特定の 2 モジュールの位置を交換する事で更なる低温化を図る手法である．

第 4 章「**計算機実験**」では，第 3 章で述べたアルゴリズムを実装し，シミュレーションを行った結果を示す．また，その結果についての考察を行う．

第 5 章「**結論**」では，本論文全体の内容を総括する．

第2章

エリート度に基づく遺伝的アルゴリズム

2.1 本章の概要

本章では本研究のベースであるエリート度に基づく遺伝的アルゴリズム [16] の特徴について説明する．まず初めにこの手法で用いるデータ構造表現であるシーケンスペアについて述べ，次に，評価値の計算及び局所改良，スケーリングとルーレット選択，そしてエリートの定義とエリート度の計算方法について説明する．更に，エリート度に基づく交叉について説明を行う．交叉方法は，エリート交叉（CPTX）と非エリート交叉（PPEX）の 2 種類がある．突然変異と局所改良についても説明を行う．

2.2 エリート度に基づく遺伝的アルゴリズムの特徴

エリート度に基づく遺伝的アルゴリズムでは，各個体毎の評価値を比較し，評価値の良いものをエリートと定義する．そして先祖にエリートを持つ個体に対して，そのエリート先祖数に応じたエリート度を与える事で，潜在的に良い構成を受け継いでいる個体の見極めを行い，それに適した交叉を行う事が出来る．

この特徴により，エリート度に基づく遺伝的アルゴリズムでは，良い局所配置を初期の世代から後期の世代まで残す事ができる．そのため，例え評価値が悪くてもエリート度が高ければ，エリート交叉により局所的な配置のみを後世に引き継いでいく事が出来るので，局所配置に大きく依存する発熱を考慮したフロアプラニングに適していると言える．

エリート度に基づく遺伝的アルゴリズムのフローを図 2.1 に示す．次節以降で，データ構造表現及び図 2.1 の各ステップについて説明を行う．

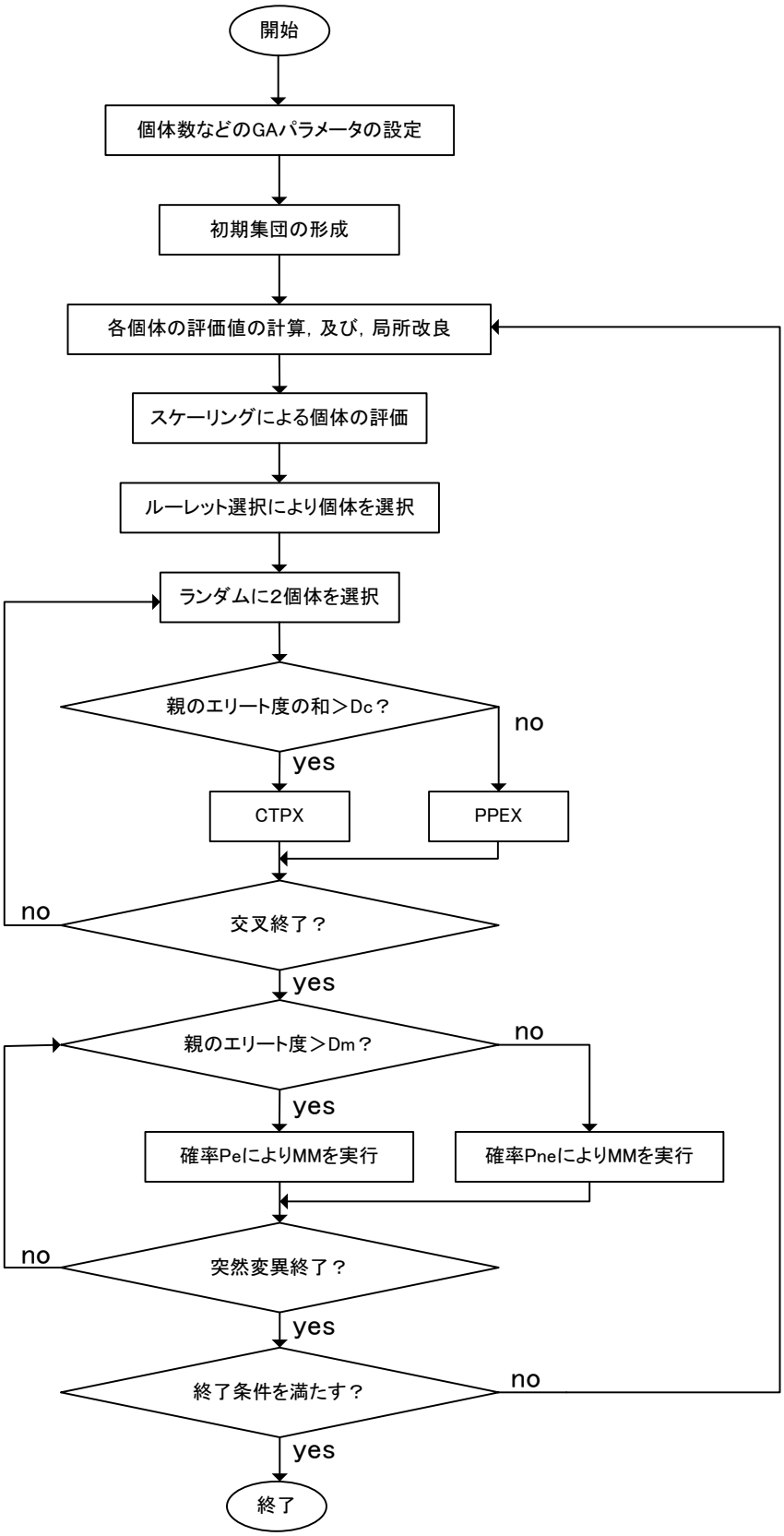


図 2.1: エリート度に基づく遺伝的アルゴリズムのフロー.

2.2.1 データ構造表現

エリート度に基づく遺伝的アルゴリズムでは，モジュールの表現法としてシーケンスペアを用いている．シーケンスペアは配置の対象となるモジュール名から構成される系列 (Γ_+, Γ_-) により方形配置を表現する方法である．各モジュール名は (Γ_+，Γ_-) のそれぞれの系列において 1 個ずつ含まれ，(Γ_+，Γ_-) の順列に基づいて任意の 2 つの方形の相対的な位置関係が指定される．

まずはフロアプランから SP を求める方法について説明する．図 2.2 に SP を求める方法を示す．図 2.2 からも分かる様に，各モジュールの左上の頂点から，チップ全体の左上に向けて線を引いていく事で Γ_+ を求める．この線は左，もしくは上にのみ進む事ができ，それぞれの線は絶対に交差してはならない．これにより全てのフロアプランに対してただ一つの Γ_+ を求める事が出来る．Γ_- もほぼ同様の方法によって求める事が出来る．

次に，SP からフロアプランを求める方法について簡単に説明する．図 2.3 にフロアプランを求める方法を示す．この図から直感的に各モジュールの相対配置を把握する事が出来る．実際にはこれをレフトダウンパッキングする事により，左下から順に当てはめていく事で図の (b) の様なフロアプランを求める事が出来る．

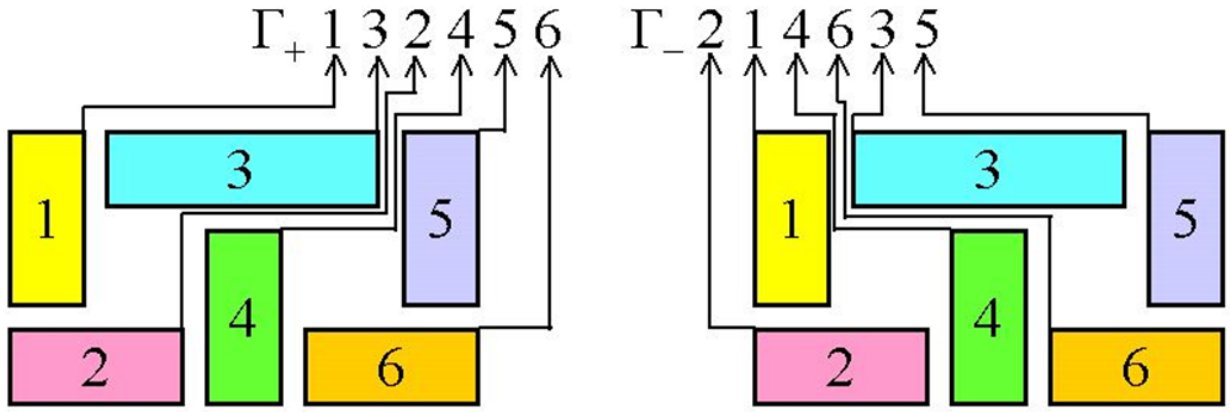
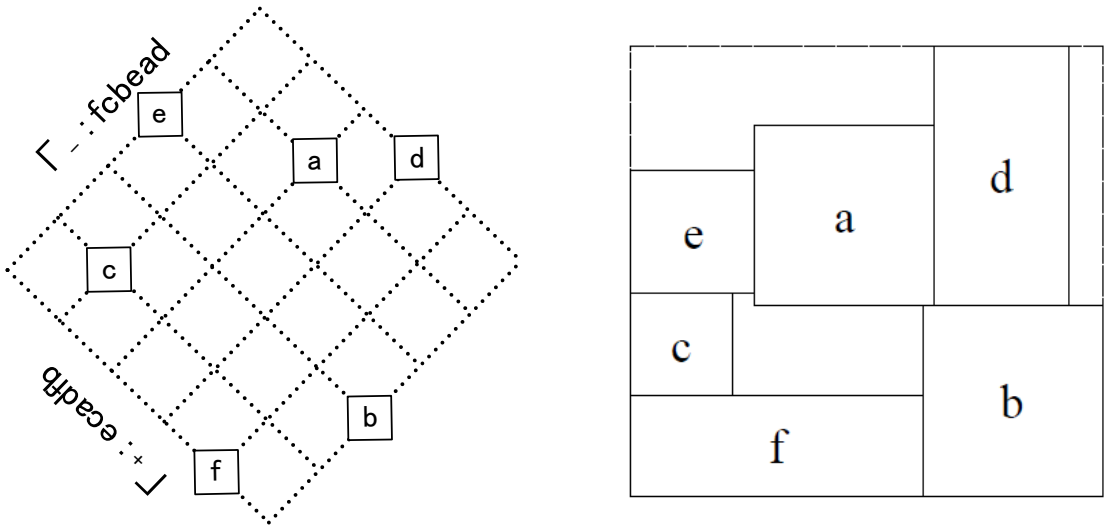


図 2.2: フロアプランから SP を求める方法.



(a) シーケンスペアの格子　　(b) 求められるフロアプラン

図 2.3: SP からフロアプランを求める方法.

2.2.2 評価値の計算及び局所改良

各個体の評価値は以下のように定義されている．

$$Cost = a \times A + b \times W$$

A，W は，面積と総配線長をそれぞれの全体平均で割った値であり，a，b は重みである．この評価値が小さい程優秀な個体であるとみなされ，次世代に残る確率が高くなる．

またこの手法では，更なる解の改善のために，局所改良を行っている．局所改良のアルゴリズムを以下に示す．

---

1. 最長経路上のモジュールを 1 個選択して向きを変えて評価値が最小となる向きに変更する．
2. 最長経路上のモジュールを 1 個選択し，Γ_+,Γ_- の位置，向きを変更し評価値の改善を行う．評価値が改善されなければ，もとの位置と向きのままにする．
3. ステップ 1，ステップ 2 を繰り返し，指定回数解が改善されない場合，又は指定回数繰り返された場合終了する．

局所改良では，世代数が進むに従い指定回数を多くする事により，前半では改良を抑え，後半では改良を強くしている．こうして効率的に解の改善を行い，回数の指定により局所解に陥らないようにしながら局所改良する事を可能としている．

2.2.3 スケーリングとルーレット選択

スケーリングとは，評価値の値を変更（小さく）して，個体間の評価値の差を実際より圧縮する操作である．スケーリングを行う事により，評価値の低い個体が淘汰される事を抑制する事が出来る．個体に多様性を持たせるため，探索の初期には頻繁に行う．

また，ルーレット選択とは評価値を用いてルーレット盤を作成する方法である．この選択方法は，単純に評価値の高いものから順に次世代に残すエリート選択法に比べて，評価値の低いものにも生き残りのチャンスがあるため，局所探索に陥る可能性が低い．ルーレット盤の例を図 2.4 に示す．図 2.4 からわかるように，評価値の高いもの程大きい面積を得る事が出来るが，評価値の低いものにもスペースが設けられているため生き残れる可能性がある．

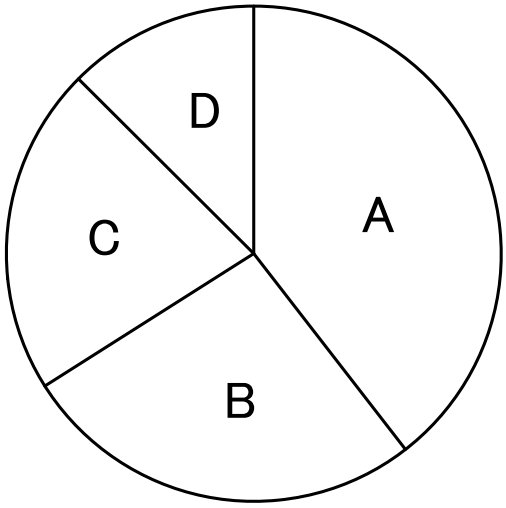


図 2.4: ルーレット盤の例.

2.2.4 エリート定義

エリート度に基づく遺伝的アルゴリズムでは，2.2.2 節で求めた評価値を用いて各個体がエリートであるかどうかを決める．世代 T における i 番目の個体（フロアプラン）x_i^T の評価値を $f(x_i^T)$ とした時に，以下の条件を満たすものをエリートと定義する．

$$f(x_i^T) \leq \mu_T - \alpha \times \sigma_T$$

なお，μ_T，σ_T は，それぞれ評価値の平均と標準偏差を表す．α はエリート決定係数であり，この値を大きくするとエリート個体数が少なくなる．

さらに，上記のエリート判別を用いて，各個体のエリート度を求める．エリート度とは，個体（フロアプラン）の潜在的な優劣を判別するための値であり，以下の式で求める事が出来る．

$$E\_\_deg(x_i^T) = \frac{\sum_{j=0}^{l_{max}} \{|Elite(x_i^T, j)| \times \beta^j\}}{\sum_{j=0}^{l_{max}} \{|Anc(x_i^T, j)| \times \beta^j\}}$$

なお，$Elite(x_i^T, j)$ は j 世代前のエリートである先祖の集合，$Anc(x_i^T, j)$ は j 世代前の先祖の集合を表す．また，l_{max} は，辿る最大先祖世代数を示す．β はエリート度影響係数であり，この値を小さくすると，先祖の影響が少なくなる．

このエリート度を基に，エリート度の高い個体同士の交叉には，親の性質を多く残す交叉手法，エリート度の低い個体同士の交叉には，解空間を広く探索する交叉手法を用いる．これにより，効率的な解の探索が出来る．

例として図 2.5 の場合を考える．図 2.5 の一番下にあるノードに注目した場合，そのエリート度は以下の式で求められる．

$$E\_\_deg = \frac{1 \times \beta^1 + 2 \times \beta^2 + 3 \times \beta^3}{1 \times \beta^0 + 2 \times \beta^1 + 4 \times \beta^2 + 6 \times \beta^3}$$

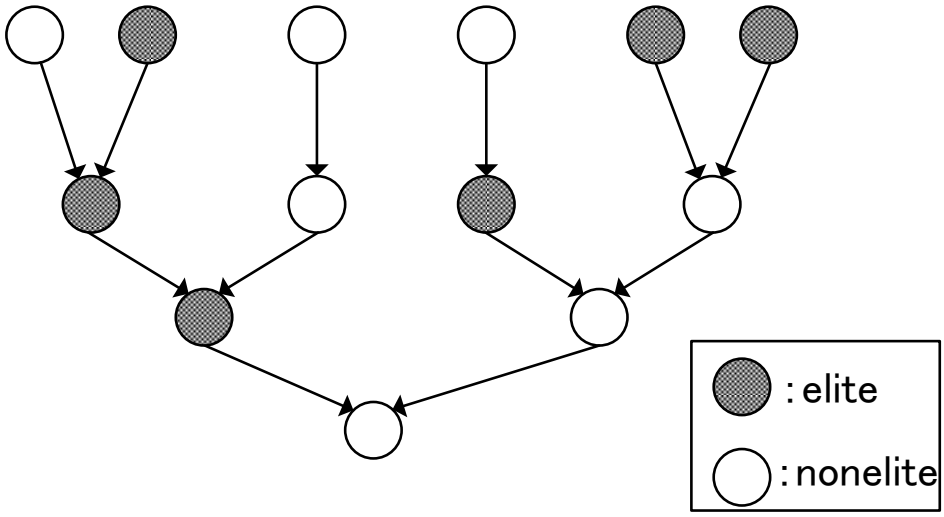


図 2.5: エリート度の計算例.

2.2.5 エリート交叉

図 2.1 からわかるように，親のエリート度の合計値が一定値以上の場合はエリート交叉を採用する．エリート交叉手法として用いられるのは，共通トポロジー保存手法（Commom Topology Preserving Crossover，CTPX）である．この手法は親の共通するモジュールの位置関係をできるだけ多く保存する手法である．具体的には，親のシーケンスの中で共通の順序を持つ最大モジュール列（Longest Common Subsequence：LCS）を保存する交叉手法である．

LCS の求め方の例として，$\Gamma_+ = \{3, 7, 5, 2, 8, 1, 10, 9, 6, 4\}$,$\Gamma_- = \{6, 2, 3, 10, 5, 7, 9, 1, 4, 8\}$ における場合を図 2.6 に示す．

図 2.6 は 4 マス毎に左下を中心と見て，そのマスの行と列のモジュール名が一致している場合には，右上マスの値に 1 足したもの，右マスの値，上マスの値の 3 つの中で最も大きいものがそのマスの目となる．モジュール名が一致していない場合は，3 マスの中で最も大きいものと同じ値となる．どちらの場合もどのマスから値をとったかを矢印で覚えておく．

上記の要領で図が完成したら，次は一番左下のマスから順に矢印を辿って値が 0 のマスまで進んでいく．その際に，通った道の中で右上に矢印を持っており，右上より値が 1 大きくなっているマスの行（または列）のモジュールが LCS となる．従って図 2.6 の場合の LCS は左下から見ていくと，4，9，5，3 である．よって，このシーケンスペアにおける LCS は逆から読んで，{3，5，9，4} となる．

CPTX では，LCS を求めたら，LCS に関しては子にそのまま遺伝させ，LCS 以外のモジュールに関しては相手親の順序に並び替えて遺伝させる．これにより，エリートの良質な個性を壊さずに遺伝させる事が出来る．

CPTX の例を図 2.7 に示す．図 2.7 では $\Gamma_,\Gamma+$ 共に LCS であるモジュールは {a,b,c,d} なので，その 4 つのモジュールに関しては親の順序を保存し，それ以外のモジュール（e と f）は相手親の順序に並び替えている．

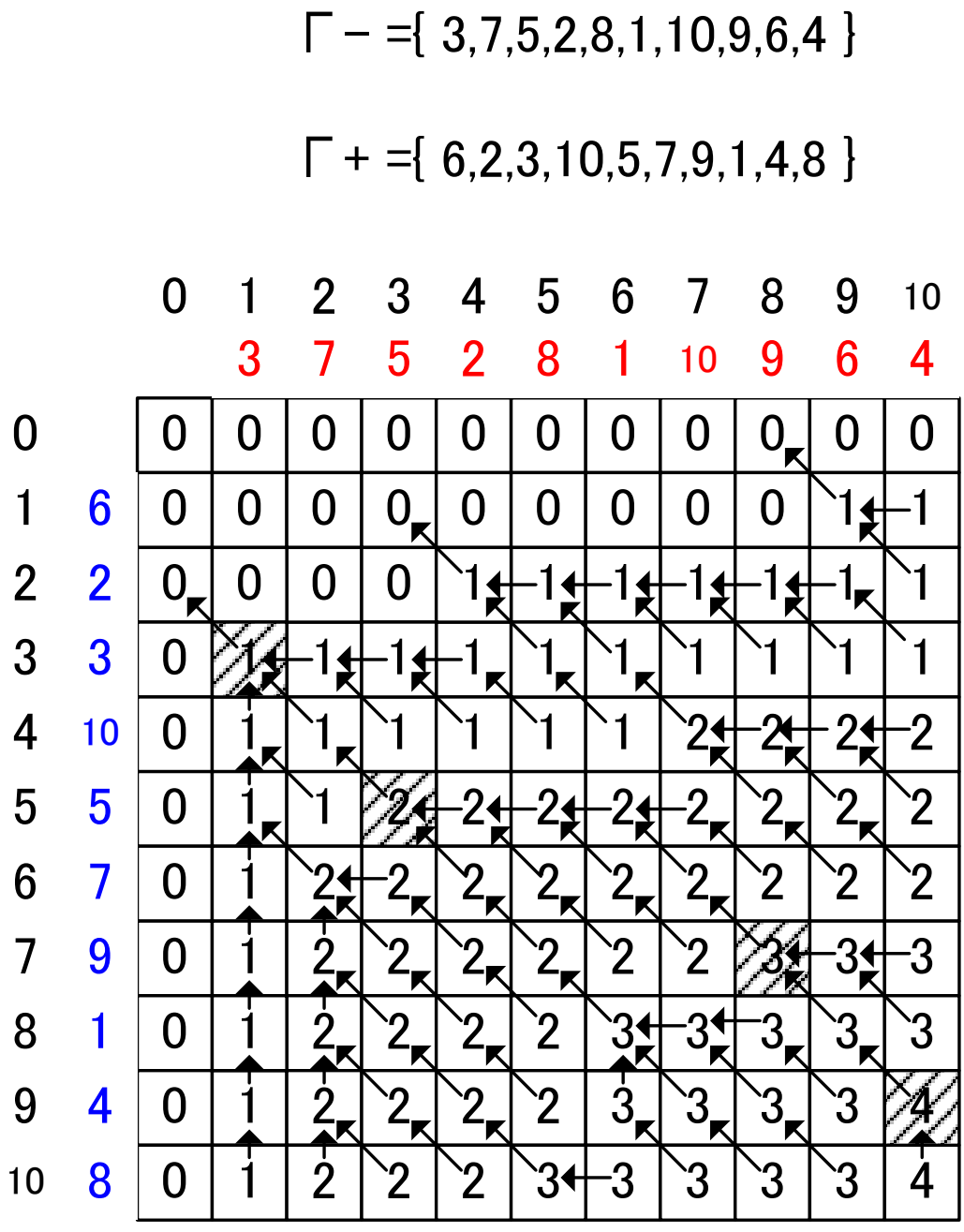

Γ − ={ 3,7,5,2,8,1,10,9,6,4 }
Γ + ={ 6,2,3,10,5,7,9,1,4,8 }
0 1 2 3 4 5 6 7 8 9 10
3 7 5 2 8 1 10 9 6 4
0 0 0 0 0 0 0 0 0 0 0 0
1 6 0 0 0 0 0 0 0 0 0 1 1
2 2 0 0 0 0 1 1 1 1 1 1 1
3 3 0 1 1 1 1 1 1 1 1 1 1
4 10 0 1 1 1 1 1 1 2 2 2 2
5 5 0 1 1 2 2 2 2 2 2 2 2
6 7 0 1 2 2 2 2 2 2 2 2 2
7 9 0 1 2 2 2 2 2 2 3 3 3
8 1 0 1 2 2 2 2 3 3 3 3 3
9 4 0 1 2 2 2 2 3 3 3 3 4
10 8 0 1 2 2 2 3 3 3 3 3 4

図 2.6: LCS 作成の例.

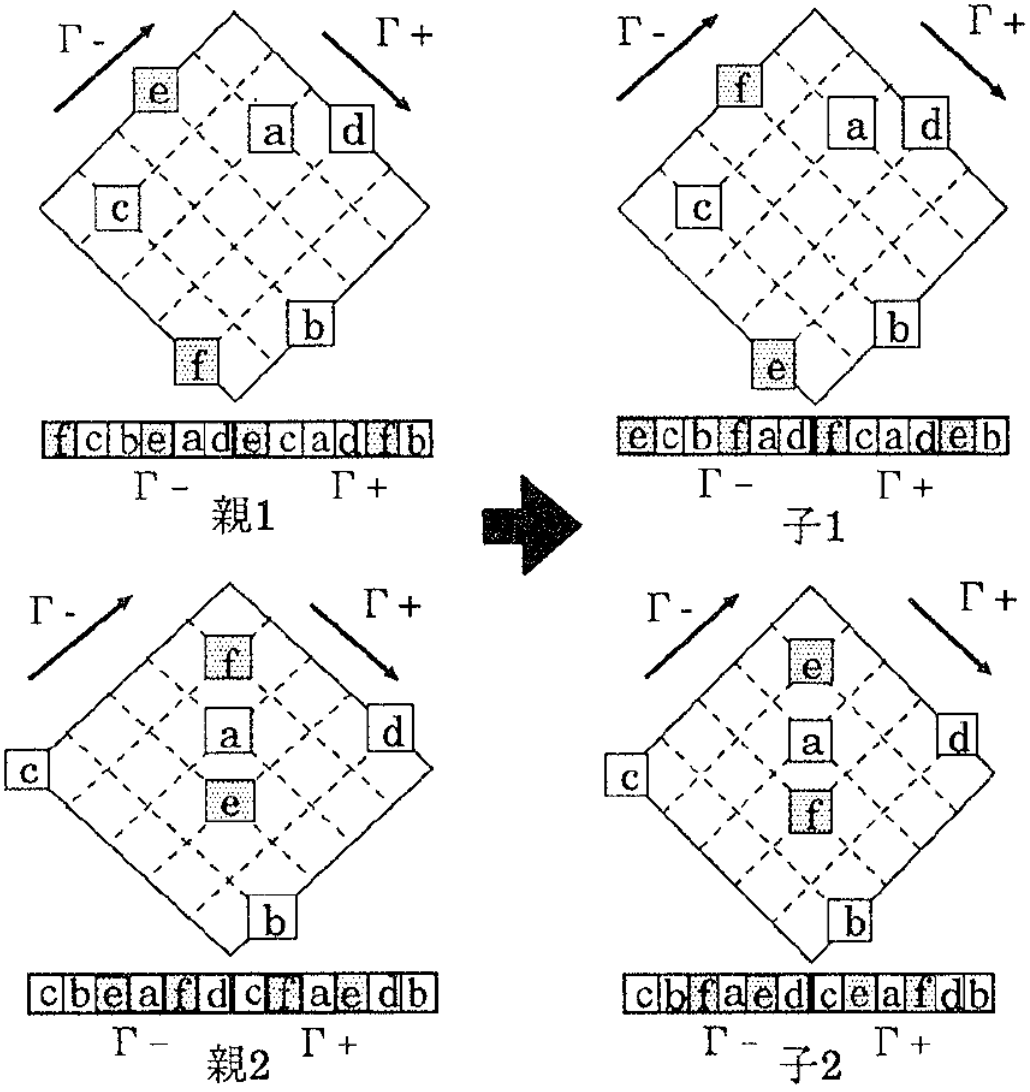

Γ -
Γ +
e
a
d
c
b
f
f c b e a d e c a d f b
Γ -
Γ +
親1
Γ -
Γ +
f
a
d
c
b
e
e c b f a d f c a d e b
Γ -
Γ +
子1
Γ -
Γ +
f
a
d
c
e
b
c b e a f d c f a e d b
Γ -
Γ +
親2
Γ -
Γ +
e
a
d
c
f
b
c b f a e d c e a f d b
Γ -
Γ +
子2

図 2.7: CTPX の例.

2.2.6 非エリート交叉

親のエリート度の合計値が一定値以下の場合は非エリート交叉を採用する．非エリート交叉手法として用いられるのは，配置依存部分交換交叉（Placement-based Partially Exchanging Crossover，PPEX）である．PPEX は，ランダムに選ばれたモジュールを中心に窓領域を作成し，窓に含まれるモジュールを交叉対象とし，対象モジュールを相手親の順序に並べ替えて遺伝させる交叉手法である．ここで言う窓とは，図 2.3(a) で表されるモジュールの相対位置を表す斜め格子上の連続した部分領域を意味する．これにより，配置上近いモジュールに関して局所的な交叉を行う事が可能となる．

PPEX の例（窓領域の 1 辺を 4 とした場合）を図 2.8 に示す．親 1 では窓領域内のモジュール a と d は，Γ,Γ+ 共に {d,a} の並びであるが，親 2 ではどちらも {a,d} の順に並んでいるため，その様に変更される．親 2 においても同様である．

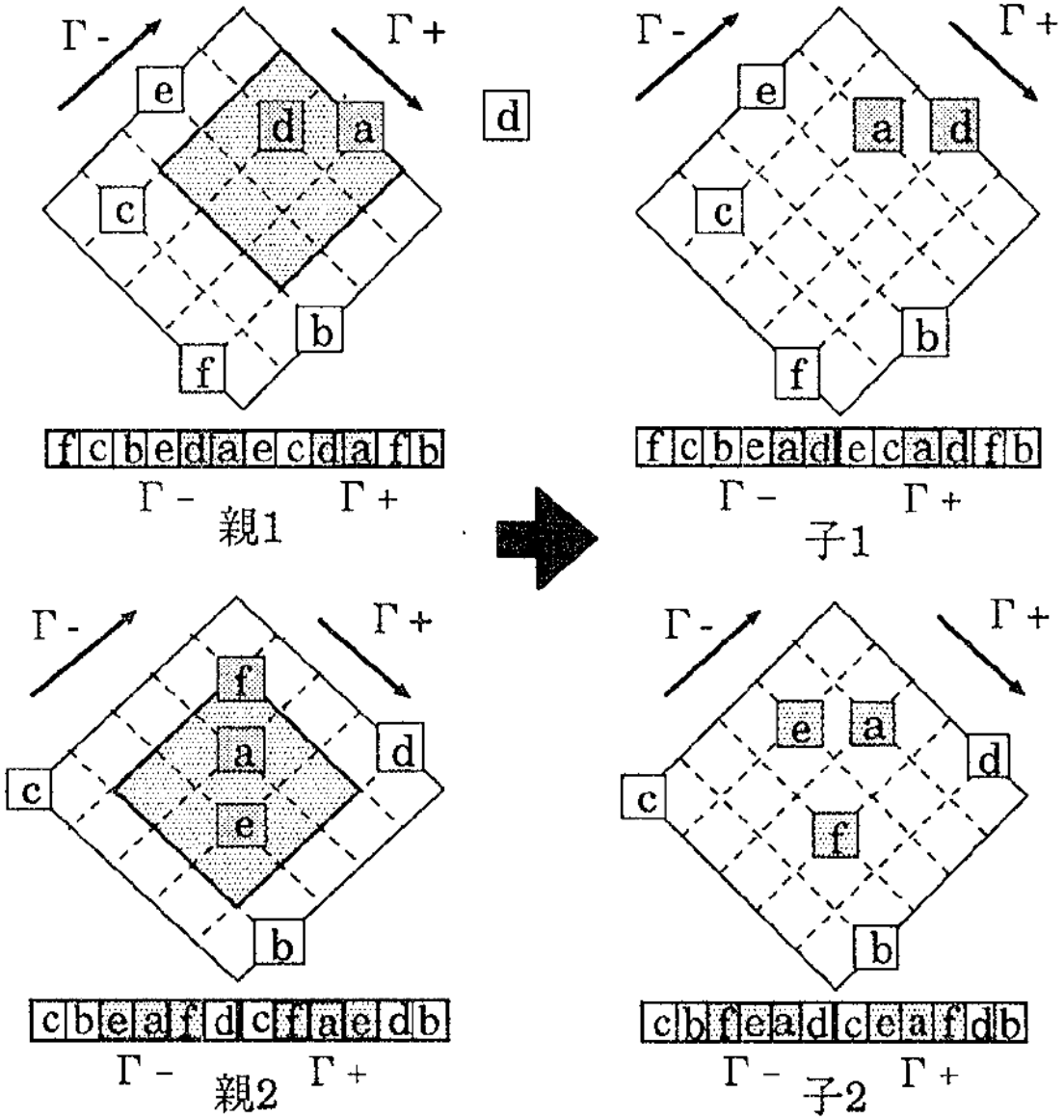

Γ -
Γ +
e
d
a
c
b
f
d
f c b e d a e c d a f b
Γ - Γ +
親1
f c b e a d e c a d f b
Γ - Γ +
子1
c b e a f d c f a e d b
Γ - Γ +
親2
c b f e a d c e a f d b
Γ - Γ +
子2

図 2.8: PPEX の例.

2.2.7 突然変異

文献 [16] では，突然変異に関しては，ランダムに選ばれたモジュール 2 つの順序と向きを交換する方法が採用されていた．突然変異の例を図に示す．

また，エリート度の高い個体に対しては低確率で，エリート度の低い個体に対しては高確率で突然変異が起こるようパラメータが設定されている．この設定により，エリート個体の良い性質を壊さないようにしている．

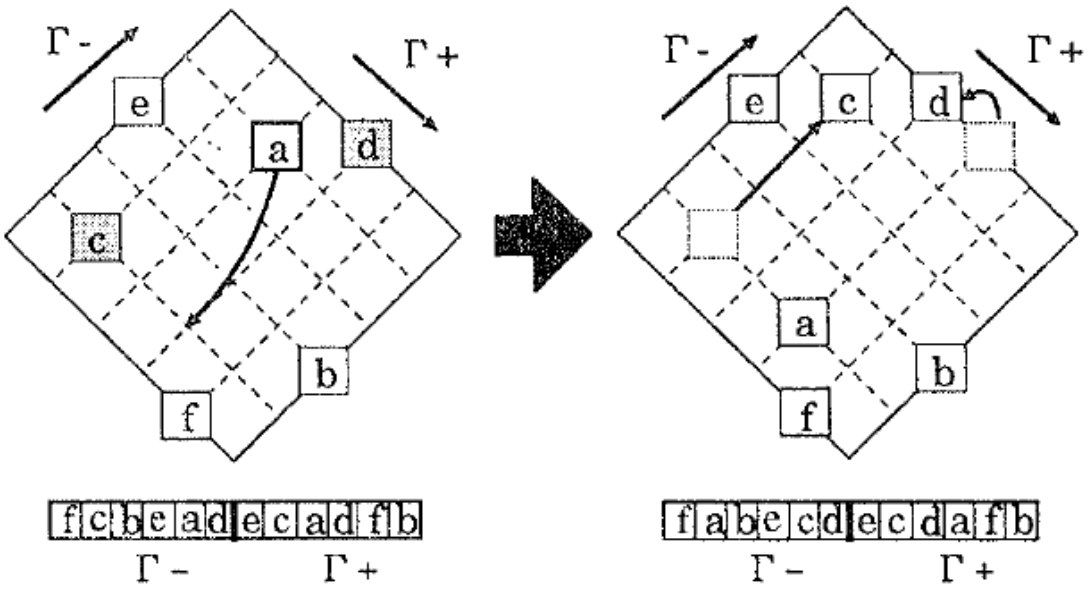


図 2.9: 突然変異の例．

2.3 本章のまとめ

本章では本研究のベースであるエリート度に基づく遺伝的アルゴリズムの特徴について説明した.

まず初めにこの手法で用いるデータ構造表現であるシーケンスペアについて述べた．シーケンスペアは配置の対象となるモジュール名から構成される系列 (Γ_+, Γ_-) により方形配置を表現する方法であり，フロアプラニングに最も良く使われる形式である.

評価関数は面積と配線を含む一般的な形式のものであり，重みである a と b の値を調整する事で設計者のニーズに沿った解を求める事が出来る．また，この手法では局所改良によって更なる解の改善を目指している．前半は改良を抑え，後半に改良回数を増やす事で局所解に陥る事を防いでいる．スケーリングも同様に，評価値の差を薄め，全ての個体に選ばれるチャンスを与えており，ルーレット選択も評価値の低い個体に生き残りの可能性を与えている.

また，エリート度に基づく遺伝的アルゴリズムで最も重要となるエリートの定義，そしてエリート度の計算方法について説明した．エリートの可否はその世代の評価値のみをもとに定められる．エリート度は先祖にエリートの個体をいくつ持つかを基準に計算される．各個体のエリート度を参考にする事で，前世から引き継いだ潜在的に優れた局所配置を後世に伝えていく事が出来る.

そして，エリート度を基にして交叉方法を決める．交叉方法はエリート交叉（CPTX）と非エリート交叉（PPEX）の 2 種類があり，CPTX では LCS モジュールを保存する事でエリートの良質な個性を壊さずに遺伝させる事が出来る．PPEX では，ランダムに決められたモジュールを中心に窓領域を生成し，領域内に含まれるモジュールを相手親の順序に並び替える方法である．これにより，配置上近いモジュールに関して局所的な交叉を行う事が可能となる.

突然変異に関しては，ランダムに 2 モジュールの位置を交換するに留まるが，エリート度の高い個体は非常に低確率で変異するという設定をする事で，エリート個体が変化してしまわないようにしている.

第3章

提案フロアプラニング手法

3.1 本章の概要

本章では，まず初めにエリート度に基づく遺伝的アルゴリズムに対する改変について説明する．エリート度に基づく遺伝的アルゴリズムは，優れた局所配置を受け継ぐ個体をエリートと定義し，エリート個体をなるべく壊さないような遺伝方法を用いているため，ホットモジュール周りの局所配置に重きを置く熱考慮フロアプラニングに適している．そこでこのアルゴリズムはそのままに，エリートとなる個体の選別方法を変えるために評価関数の改変を行う．改変後の評価関数には当然温度という新要素が含まれるため，本章では動作温度の計算方法に関しても，既存の方法と，より優れた新規手法について説明を行う．また，局所改良に関しても若干の改変を行い，各個体の温度を下げるような手法とする．

次に，提案手法の 2 つ目として，GA によって出力されたフロアプランに対するモジュール置き換え手法について説明する．モジュール置き換え手法は，特定の 2 モジュールの位置を交換する事で更なる低温化を図る手法である．

3.2 エリート度に基づく遺伝的アルゴリズムの改変

本節では，エリート度に基づく遺伝的アルゴリズムに対する改変について説明する．第 1 章でも述べたが，エリート度に基づく遺伝的アルゴリズムは温度を考慮していない．そのため，評価関数に動作温度を加える必要がある．動作温度は，いくつかのモジュールを監視対象とし，そのモジュール郡の熱移動の合計値を消費電力密度による近似で求める．この際の監視対象モジュール郡の算出方法については 3.2.1 節で既存の方法及の説明及び，より優れた新規手法の提案を行う．また，温度に注目したいので，評価関数の要素のうちで，不要なものは取り除く必要がある．提案手法では，面積を評価関数から取り除き，アウトライン制約として与えている．局所改良は，ホットモジュールに隣接するモジュール郡に対して回転操作を行う事で対象ホットモジュールの温度を下げるという手法である．

3.2.1 動作温度の計算方法

評価関数の要素としての動作温度は，いくつかのモジュールを監視対象とし，そのモジュール郡の熱移動の合計値を消費電力密度による近似で求めた値で表す．

まず初めに隣接する 2 モジュールの熱移動について説明する．隣接する 2 モジュール間の熱移動の量は，2 モジュール間の温度差と 2 モジュールが接している部分の長さ (shared__length) に比例するため，図 3.1 のようにそれぞれの温度が T_1，T_2 であるモジュール間の熱移動 H は以下の式で表される [10]．

$$H(T_1, T_2) = (T_1 - T_2) \times shared\_\_length$$

また，各モジュールの定常状態での動作温度 T は，消費電力 P と熱抵抗 R を用いて $T = P \times R$ で表される．これに $R = \frac{t}{k \times A}$（t は材質の厚さ，k は熱伝導率，A は面積）を代入すると，以下のようになる．

$$T = P \times (\frac{t}{k \times A}) = (\frac{P}{A}) \times (\frac{t}{k}) = (\frac{t}{k}) \times d$$

d は電力密度で，消費電力÷面積で表される．

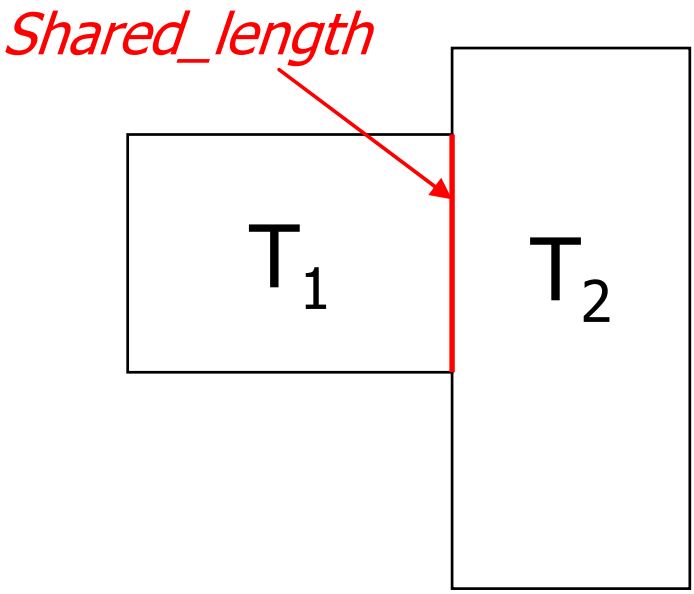


図 3.1: 動作温度の計算例．

前述の式より，$H(T_1, T_2)$ は $H(d_1, d_2)$ に近似する事が出来る．故にモジュール間の熱移動は電力密度の差と接地面の長さから求める事が出来る．

文献 [10] の手法では，電力消費の激しいモジュールを数個（1〜4）選んで，それらの熱移動の合計を観察し，その値を温度評価値 T とする．式で表すと以下のようになる．

$$H(d) = \sum H(d, d_i)$$

しかし，モジュールの発熱量は自身の熱量と隣接モジュールの熱量という 2 つの要因によって決まるため，自身の消費電力密度だけを基準に監視モジュールを決めるべきではないと考える．そのため提案手法ではモジュールの周の長さと消費電力密度の 2 つの側面から監視モジュールを決定する．具体的には，以下の式で求める C がより高いものを監視モジュールとする．なお，x，y はそれぞれの重みである．

$$C = x \times （周の長さ） + y \times （消費電力密度）$$

このような計算方法を用いる事で，消費電力密度が比較的高く，周が長いため多くのモジュールと隣接するであろうモジュールを選び出し監視する事が出来る．

3.2.2 評価関数の設定

文献 [10] では以下の評価関数を用いて個体の評価を行っている．

$$Cost = a \times A + b \times W + c \times T$$

なお，A，W は，面積と総配線長をそれぞれの全体平均で割った値であり，T は前節で説明した文献 [10] の計算方法で算出した値を世代の平均値で割った値である．また，a，b，c は重みである．

しかしこれでは温度に対する重要度が単純に考えると 1/3 になってしまうため，本研究の目的から遠ざかってしまう．そのため，本手法では固定アウトライン（FixedOutline）制約を用いたフロアプラニングを行う事で面積を既存の評価値から除外し，評価関数における温度の重要度を上げる事とする．

固定アウトライン制約について説明する．アウトラインとは全モジュールを包括する最小矩形のラインの事であり，アウトライン制約とは，そのラインを制約として与えるという意味である．アウトライン制約の例を図 3.2 に示す．図 3.2 の右側が制約を満たさない場合で，左側が制約を満たす場合である．

アウトラインの高さ H_* と幅 W_* は以下の式で求める [5]．

$$H_* = \sqrt{(1+\gamma)Aratio}, W_* = \sqrt{(1+\gamma)A/ratio}$$

なお，A はモジュールの合計面積（mm^2），γ はデッドスペースの最大許容割合（%）を表す．また，$ratio$ は H_*/W_* で表される縦横比 (aspect ratio) を表す．

固定アウトライン制約を考慮する場合，制約を満たさない個体を全て排除して探索を進めると局所探索に陥り，最適解にたどり着けない可能性がある．そのため，制約を満たさない

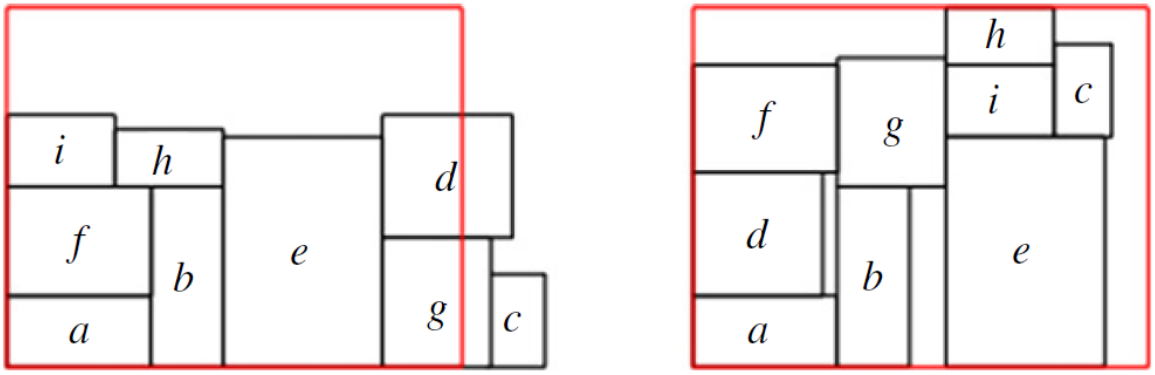


図 3.2: アウトライン制約の例.

個体でも，良い配置構成を保持している可能性があり，次世代に残す必要がある [5]．上記の理由より，本手法ではルーレット選択が終了した後で，選択されたがアウトライン制約を満たさない個体の中から評価値の低い個体を順に一定数取り出す．そしてルーレット選択では選ばなかった個体の中から，制約を満たし，かつ評価値の高い個体も一定数取り出し，前者と後者を交換する．これにより，制約は満たさないが高い評価値を持つ個体と，評価値はそこまで高くないが制約を満たす個体が次世代に残る事となる．このように世代を重ねる事で，最終的には制約を満たし，かつ評価値の高い個体が生まれる．

面積を関数から除外したら，次は動作温度 T を文献 [10] の計算方法から，前節で提案した計算方法で求めたものに変える．更に，温度に関係する要素として，監視モジュールに接するモジュール数という値を評価関数に加える．今まで説明してきたように，動作温度の計算時には監視モジュール郡の熱移動に着目し，その合計値が最も大きくなる場合を探すため，監視モジュール郡の周りに消費電力密度の小さいモジュールが多数隣接する様な配置が GA の次世代に選ばれやすい．しかし，大抵の場合，それら隣接モジュールの中の 1 つや 2 つは，実際には他の影響により大きく発熱してしまう．そのため，監視モジュール郡に接するモジュールの数も評価要素として考慮する必要がある．

以上の考えより，私の提案手法では以下の評価関数を用いる事とする．

$$Cost = a \times W + b \times \acute{T} + c \times L$$

なお，$\acute{T}$ は前節で提案した方法で求めた温度を，L は監視モジュールに接するモジュール数を，それぞれ世代の平均値で割った値である．また，a，b，c は重みである．

3.2.3 局所改良

ホットモジュール周りの隣接状況は非常に重要であるため，局所改良によって改善を行う必要がある．そのため，アウトライン制約を満たす個体に関しては，その個体の動作温度がより低くなるように局所改良を行う．アウトライン制約を満たす個体に対する局所改良のフローを図 3.3 に示す．この局所改良により，ホットモジュールに隣接するモジュールは，出来るだけ接地面が小さくなるような隣接の仕方をするため，ホットモジュールの温度が下がる．

また，アウトライン制約を満たさない個体に対しては，まずは枠内に収まる事が先決なので，2.2.2 節で述べた従来の局所改良法を適用する．

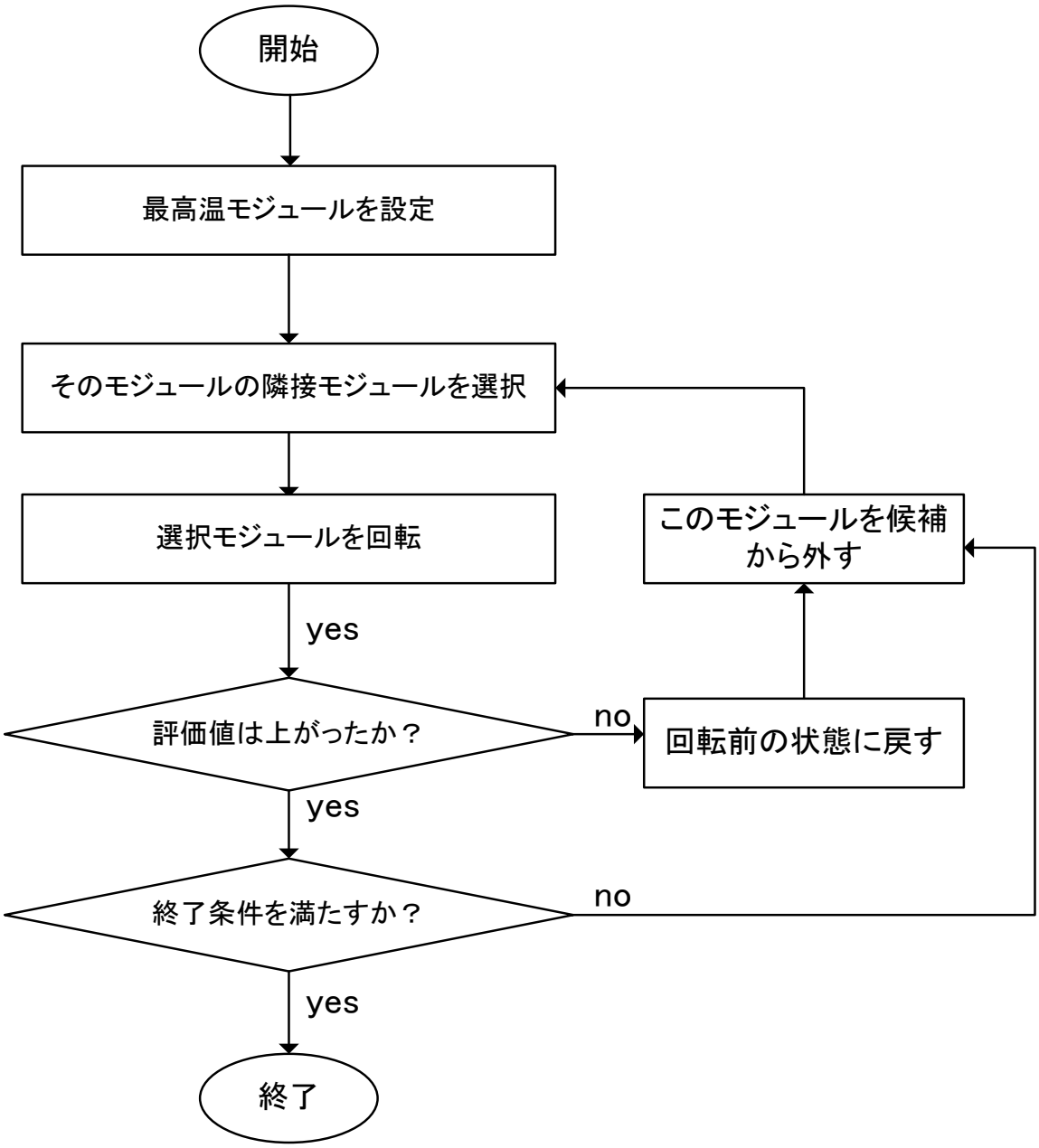


図 3.3: アウトライン制約を満たす個体に対する局所改良法.

3.3 モジュール置き換え手法

本節では，提案手法の 2 つ目として置き換え手法について説明する．置き換え手法とは GA で求めたフロアプランに対して，更に局所的にモジュールの置き換えを行う事によって最高温度を下げる手法である．エリート度に基づく GA では狙って特定の 2 モジュールを交換する事が出来ないため，GA 終了後にこの様なアプローチを行う必要があると考えられる．

3.3.1 置き換え対象

この手法では以下の 2 つのモジュールを対象として置き換えを行う．

- 置き換えたいモジュール：
  ホットモジュールに隣接する中で最も温度の高いもの
- 置き換え先：
  置き換え可能な候補の中で，最も消費電力密度の低いもの

何故上記の 2 モジュールを対象とするのか説明する．まず初めに，置き換えたいモジュールだが，これはホットモジュールではなく，それに隣接する中で最も温度の高いモジュールとする．ホットモジュールは主に 3.2.1 節で述べた監視モジュールであるが，その位置は GA の探索により既に適位置にあるものと考えられる．また，ホットモジュールはそれ自体が強く発熱するため，移動させると移動先でも発熱してしまい，本末転倒になってしまう．そのため，ホットモジュールの位置は移動させずに，その周りにあるモジュールを移動させる事が結果的にホットモジュール自身の温度を下げる事に繋がるのである．そして，隣接モジュールの中でどれを移動させるのかと言えば，それはもちろん最も温度の高いものとなるわけである．

次に置き換え先だが，大きさがあまりにも違うモジュール同士は交換出来ないため，必然的にいくつかの候補に絞られる．候補の選び方は次節で詳しく説明するが，実際に交換するのは最も低温ではなく，最も消費電力密度の低いモジュールとする．この理由は，置き換えた時に低温になるのはどちらかと考えればわかる．低温なモジュールは自分自身の発熱量が少ないのか周りによって冷やされているのか判別がつかない．しかし，最も消費電力密度の低いモジュールは，前者に比べて多少温度は高くても，自身の発熱量は少ないため，移動先では前者よりも低温になる可能性が高い．そのため，交換するのは前者ではなく後者の，最も消費電力密度の低いモジュールとするのが適していると考えられる．

次節で置き換え候補の選び方，実際の置き換え方について説明する．

3.3.2 置き換えアルゴリズム

置き換えアルゴリズムを以下に示す.

1. 置き換え対象のうち，面積の大きい方を小さい方に置き換える.
   入りきらない場合は，大きい方のモジュールを中心にスペースを空ける.
   それでも入りきらない場合は失敗となる.
2. もとの小さい方のモジュールを中心にスペースを空ける.
3. 小さい方に大きい方を重ねる.
   入りきらない場合は失敗となる.

上記の 3 ステップにより置き換えは完了するが，前節で述べたように置き換える 2 つのモジュールはなんでも良いわけではない．置き換えたいモジュール，つまりホットモジュールに隣接する中で最も温度の高いモジュールが決定したら，それと置き換える事が出来るモジュールの候補を見つける必要がある．その場合に用いるのが，上記のアルゴリズムである．ステップ 3 において置き換えが成功するモジュールを保存しておき，それを置き換え候補とするのである．そして置き換え候補の中から最も消費電力密度の低いものを選んで，先程選んだ置き換え希望モジュールとの置き換えを実行する.

ステップ 1,2 で出てきた，スペースを空けるという操作について説明を行う．モジュールを置き換えようとすると，両者の大きさの違いから当然スペースが足りないという場合が生じてくる．その場合に，前述したスペースの確保という操作が必要となってくる.

この操作は単純で，中心とするモジュールの中心の座標を求め，その座標とその他全てのモジュールの中心座標を比べ，その相対位置関係を把握する．そして右上にあるモジュールは更に出来る限り右上へ，同じく左上にあるモジュールは出来る限り左上へ，右下左下に関しても同様に，アウトラインを超えない範囲で移動を行う．これにより中心と定めたモジュールの周りにスペースが出来る.

実際に置き換えアルゴリズムのステップ 1 から 3 を行うとフロアプランがどの様に変化するかは，次章で実験結果をもとに説明する.

3.4 本章のまとめ

本章では，まず初めにエリート度に基づく遺伝的アルゴリズムに対する改変について説明した．改変は，動作温度の計算方法と評価関数の設定方法，局所改良法について行った．

動作温度は，いくつかのモジュールを監視対象とし，そのモジュール郡の熱移動の合計値を消費電力密度による近似で求めるのだが，その際の監視対象を消費電力密度だけでなくモジュールの周長も考慮して選択する手法を提案した．

評価関数に関しては，動作温度にフォーカスするために，面積は評価関数から除外し制約として与えた．また，ホットモジュールに必要以上にモジュールが隣接しないように，隣接モジュール数を評価関数の要素として加えた．

局所改良では，ホットモジュールに隣接するモジュールに対して回転操作を行う事で，出来るだけ接地面が小さくなるような隣接の仕方をさせて，ホットモジュールの温度を下げる手法を提案した．

また，本章では GA によって出力されたフロアプランに対して行う，モジュール置き換え手法について提案を行った．この手法は，ホットモジュールに隣接する中で最も温度の高いモジュールと，置き換え可能な候補の中で，最も消費電力密度の低いモジュールの位置を交換する事で更なる低温化を図る手法である．

第4章

計算機実験

4.1 本章の概要

本章では，第 3 章で述べたアルゴリズムを実装し，シミュレーションを行った結果を示す．

まず始めに，実験の前準備として，前提条件，入出力などの各種の値，比較対象，シミュレーションを行う際の実験環境について説明する．そして実際にシミュレーションを行い，その結果について考察を行う．また，3.3 節で述べた置き換えアルゴリズムについても，各ステップにおける変化の様子を例を用いて説明する．

4.2 実験の準備

本節では実験を行うための各種設定について説明する．

前提条件

提案手法における前提条件を以下に示す．

- モジュールは全てハードモジュールであるとする．
  各モジュールは面積と共に縦横の長さも決まっているため，回転は可能だが変形は出来ない．
- 配線は内部接続行列（interconnect matrix）と HPWL（Harf Perimeter Wire Length）を用いて求めるものとする．
  内部接続行列によってモジュール間の接続の有無と配線本数を調べ，HPWL によってその長さを求める．HPWL の例を図 4.1 に示す．
- チップ全体の面積は，全モジュールを包括する最小矩形の面積であるとする．
- 評価関数における動作温度 T は，3.2.1 節で述べたが，監視モジュール郡の熱移動量を消費電力密度による近似により求めた値である．
  実際に生成されたフロアプランの熱を評価する際には，後述するシミュレータを使用する．

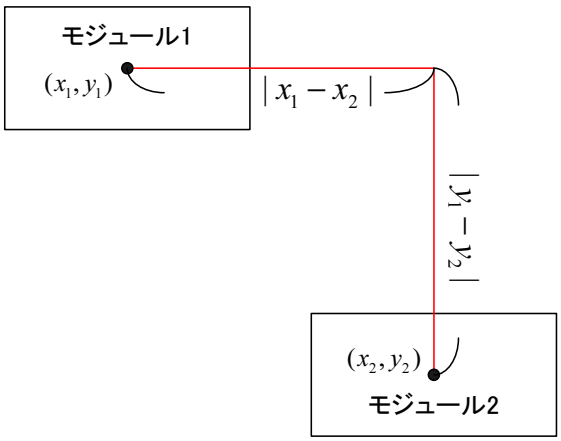


図 4.1: HPWL の例．

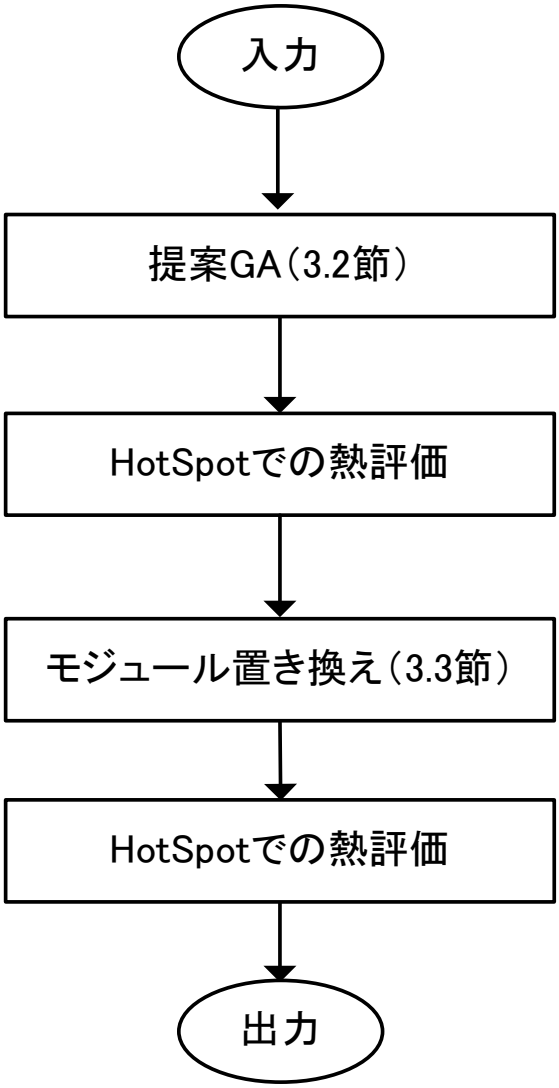


図 4.2: シミュレーションの手順.

各種データ

シミュレーションの手順を図 4.2 に示す．まずは入出力について述べる．入力データを以下に示す.

- モジュール情報.
  各モジュールは x 座標，y 座標，幅，高さ，消費電力密度を持っている.
- 内部接続行列.

上記の入力データの内，モジュールの幅，高さと内部接続行列はベンチマークから取得する．座標は入力時は全て（0,0）であり，GA による探索で実際の値を得る．また，消費電力密度はランダムに与える.

表 4.1: 提案 GA の設定パラメータ.

・世代数	2000
・個体数	500
・交叉確率	0.6
・エリート突然変異確率	0.01
・非エリート突然変異確率	0.05
・エリート度影響係数（β）	0.5
・エリート決定係数(α)	0.2
・最大局所改良回数	50

出力データを以下に示す.

- 生成されたフロアプラン.
  フロアプランは面積，総配線長，最高温度を持っている.
- モジュール情報.
  各モジュールの動作温度も出力される.

図 4.2 中にもあるようにフロアプランの熱評価は 2 回行うが，その際には HotSpot4.2 という熱計算シミュレータ [17] を使用する.

次に実験で用いるパラメータについて説明する．提案 GA で用いる各種パラメータを表 4.1 に示す．アウトライン制約時の縦横比 $ratio$，空白許容率 γ をそれぞれ 1.0，15 % とする．また，3.2.1 節で述べた監視モジュール決定のためのコスト関数で使用するパラメータ x，y はそれぞれ 0.6，0.4 とする.

比較対象

次に比較対象について説明する．本章では，以下の 4 手法を比較する.

1. 従来手法：文献 [16] のエリート度に基づく遺伝的アルゴリズム.
   $Cost = a \times A + b \times W (a = 0.3, b = 0.7)$
2. 従来手法＋アウトライン制約＋温度評価：従来手法の評価関数から面積を取り除き，文献 [10] の方法で計算した温度を加えたもの.
   面積はアウトライン制約として与え，パラメータは前述の通りである.
   $Cost = a \times W + b \times T (a = 0.3, b = 0.7)$

3. 提案 GA：エリート度に基づく遺伝的アルゴリズムにおいて，動作温度の計算，評価値の計算，局所改良法の 3 つに改変を加えたもの．
   $Cost = a \times W + b \times \acute{T} + c \times L (a = 0.2, b = 0.4, c = 0.4)$
4. 提案 GA ＋モジュール置き換え手法：本論文で提案する発熱量の低減を目的としたフロアプラニング．
   評価関数は上と同じである．

実験ではこれら 4 つの手法を比較する事で提案手法の有用性を証明する．

実験環境

実験環境は，OS が Debian GNU/Linux，CPU が Intel Xeon 3.40Ghz，メモリ容量 4GB である．実験ではベンチマークとして MCNC ベンチの ami33，及び ami49 を使用する．エリート度に基づく遺伝的アルゴリズムを初めとするプログラムは全て C 言語で実装を行った．一回のシミュレーションにかかる時間は約 1 時間である．

4.3 結果と考察

本節では実験を行った結果を示し，考察を行う．まずは，前節で比較対象として挙げた 4 つの手法の中で，上 3 つについて実験結果を示す．ami33 のフロアプランをそれぞれ図 4.3, 4.4，4.5 に示す．

従来手法ではモジュール 18 が最高温度で 77.6 ℃，従来手法＋アウトライン制約＋温度評価の結果ではモジュール 15 が 75.8 ℃，提案 GA ではモジュール 14 が 73.6 ℃となった．

従来手法は温度を考慮していないため，消費電力密度の高いモジュール 18 が中心に配置され高温になっている．従来手法＋アウトライン制約＋温度評価では，監視モジュールに関しては温度を抑える事が出来ているが，その代わりに比較的消費電力密度が高く面積も大きいモジュール 15 と 13 が隣同士に配置されてしまっている．一方，提案 GA では，18,13,5 と

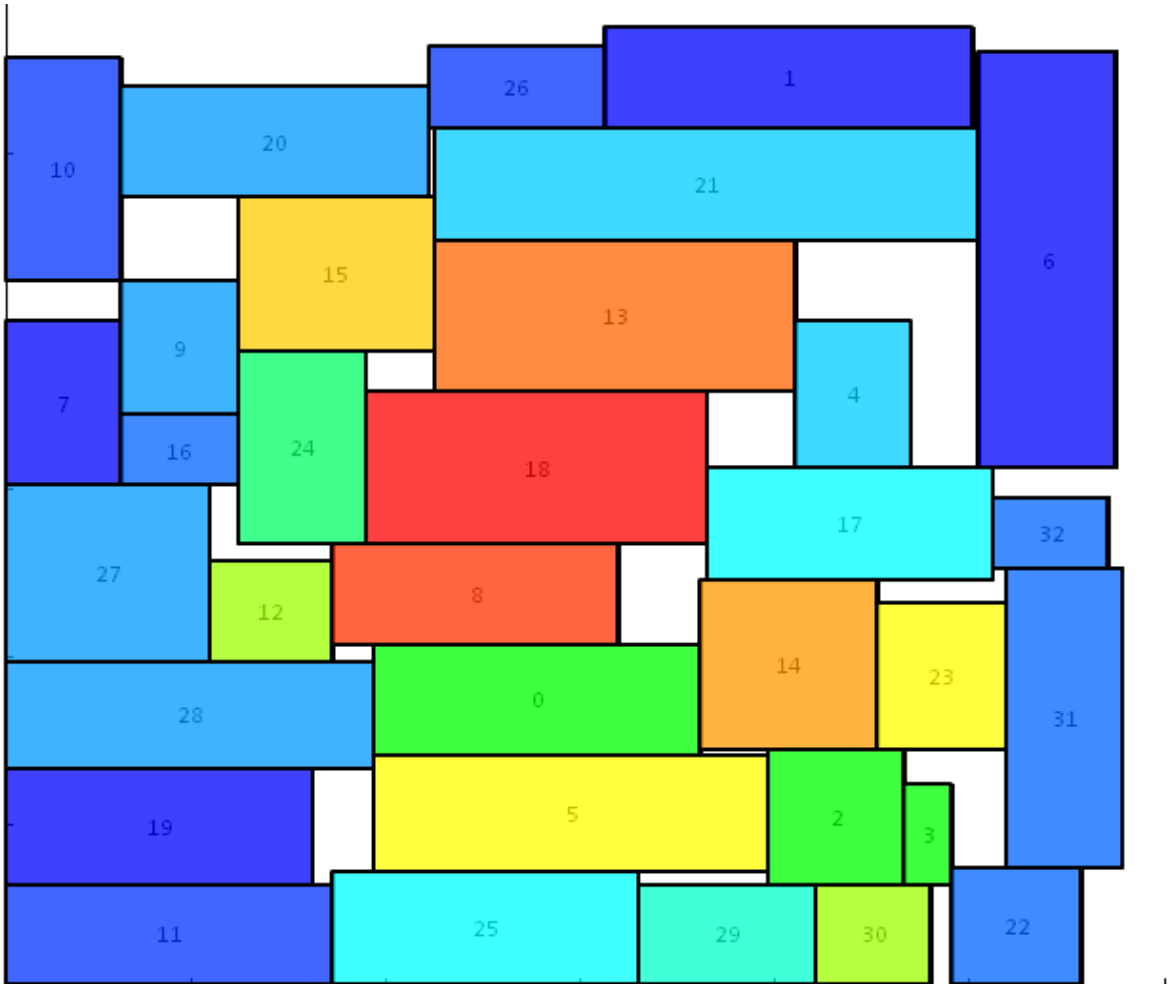


図 4.3: ami33 における従来手法の結果.

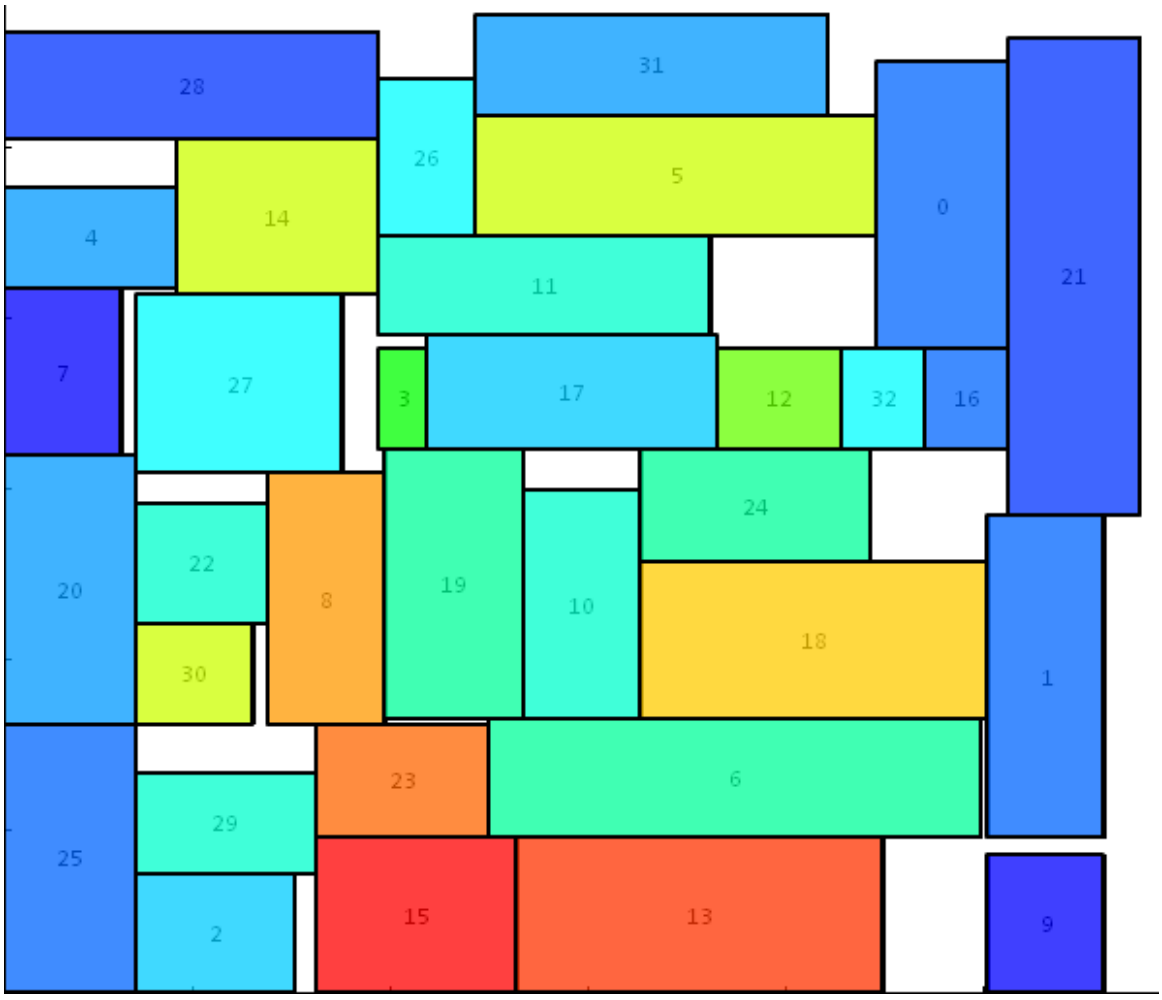


図 4.4: ami33 における従来手法＋アウトライン制約＋温度評価の結果.

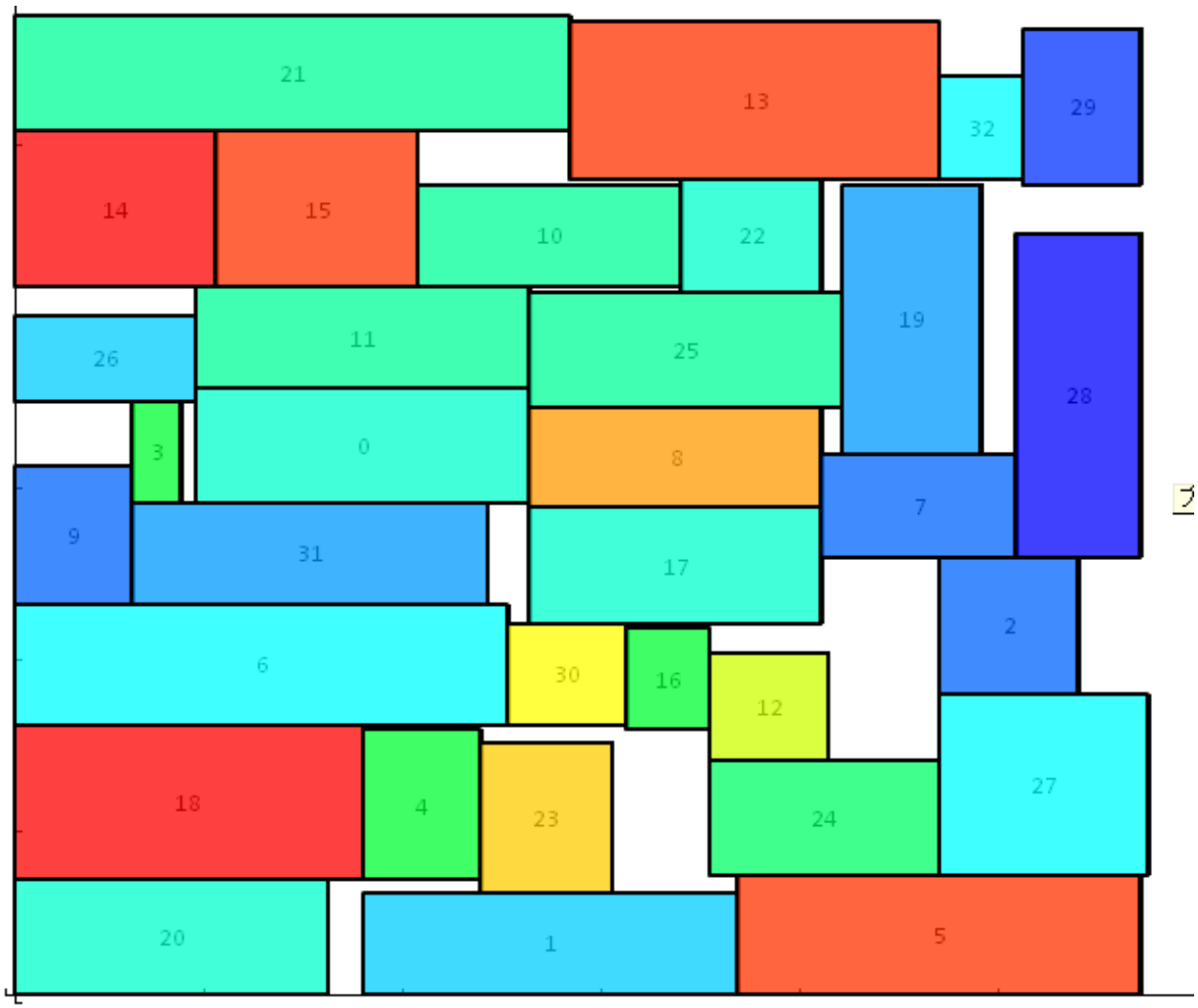


図 4.5: ami33 における提案 GA の結果.

表 4.2: ami33 における 4 つの手法のシミュレーション結果.

	面積(mm²)	総配線長(mm)	最高温度(℃)
従来手法	1.29	57.8	77.6
従来＋制約＋温度評価	1.31	84.5	75.8
提案GA	1.30	77.7	73.6
提案GA+置き換え	1.30	83.9	72.9

表 4.3: ami49 における 4 つの手法のシミュレーション結果.

	面積(mm²)	総配線長(mm)	最高温度(℃)
従来手法	40.12	1063.6	80.6
従来＋制約＋温度評価	40.30	1107.5	76.4
提案GA	39.87	1120.2	75.7
提案GA+置き換え	39.78	1143.3	74.8

いう比較的消費電力も高く面積も大きいモジュールを監視する事が出来ている．また，これら監視モジュール郡に接するモジュール数を評価関数に組み込んでいるため，監視モジュールは全て隣接数が少なくなるよう外周部分に位置どっている事がわかる．これらの働きにより，提案 GA ではチップ全体をより低温に保つ事が出来ている．

モジュール置き換え手法について説明する．アルゴリズム自体は 3.3.2 節で説明しているので，本節ではアルゴリズムの各ステップでフロアプランがどの様に変化しているのかを図 4.5 の結果をもとに説明する．

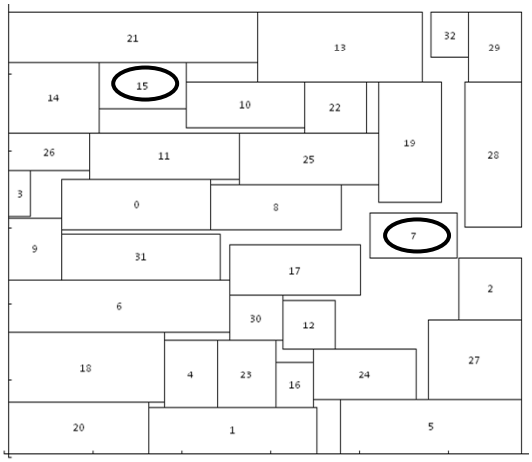
図 4.5 において，ホットモジュールであるモジュール 14 に隣接する中で最も高温であるモジュール 15 を置き換え対象の 1 つ目とし，候補の中で最も消費電力密度の低いモジュールであるモジュール 7 をもう 1 つの対象とする．実際にモジュール 15 と 7 を交換する様子を図 4.6 に表す.

また，この置き換え手法を監視モジュール 3 つに対して行った結果を図 4.7 に示す．この時の最高温度はモジュール 5 で 72.9 ℃となった.

今まで紹介した 4 つの手法の結果を表 4.2 に纏める.

また，ami49 における 4 手法のフロアプランを図 4.8 から 4.11 に，纏めを表 4.3 に示す.

1. 置き換え対象のうち，大きい方を小さい方に置き換える.



2. もとの小さい方のモジュールを中心にスペースを空ける.

3. 小さい方に大きい方を重ねる.

図 4.6: モジュール置き換えの様子.

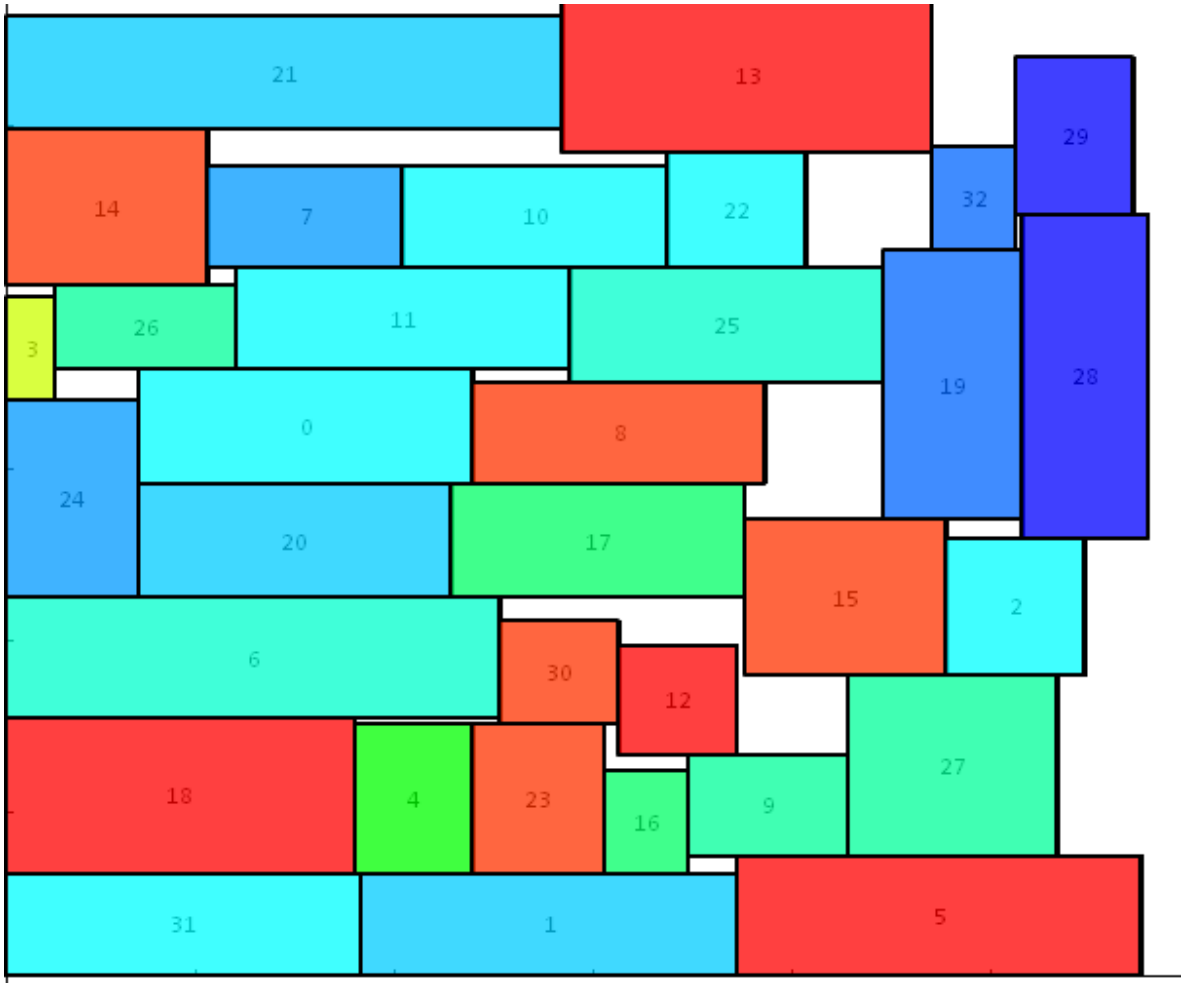


図 4.7: ami33 における置き換え手法の適用結果（3 回分）.

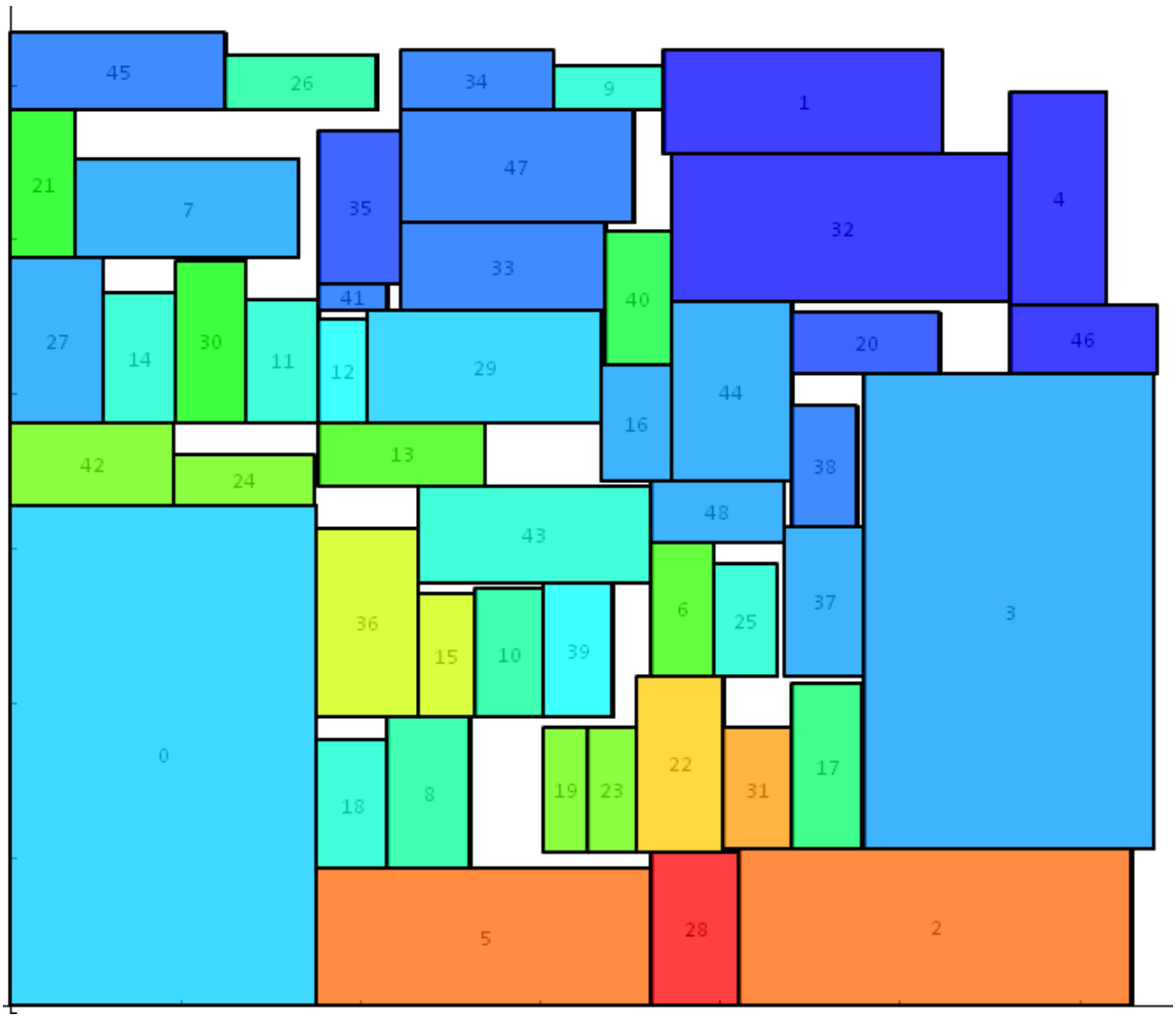


図 4.8: ami49 における従来手法の結果.

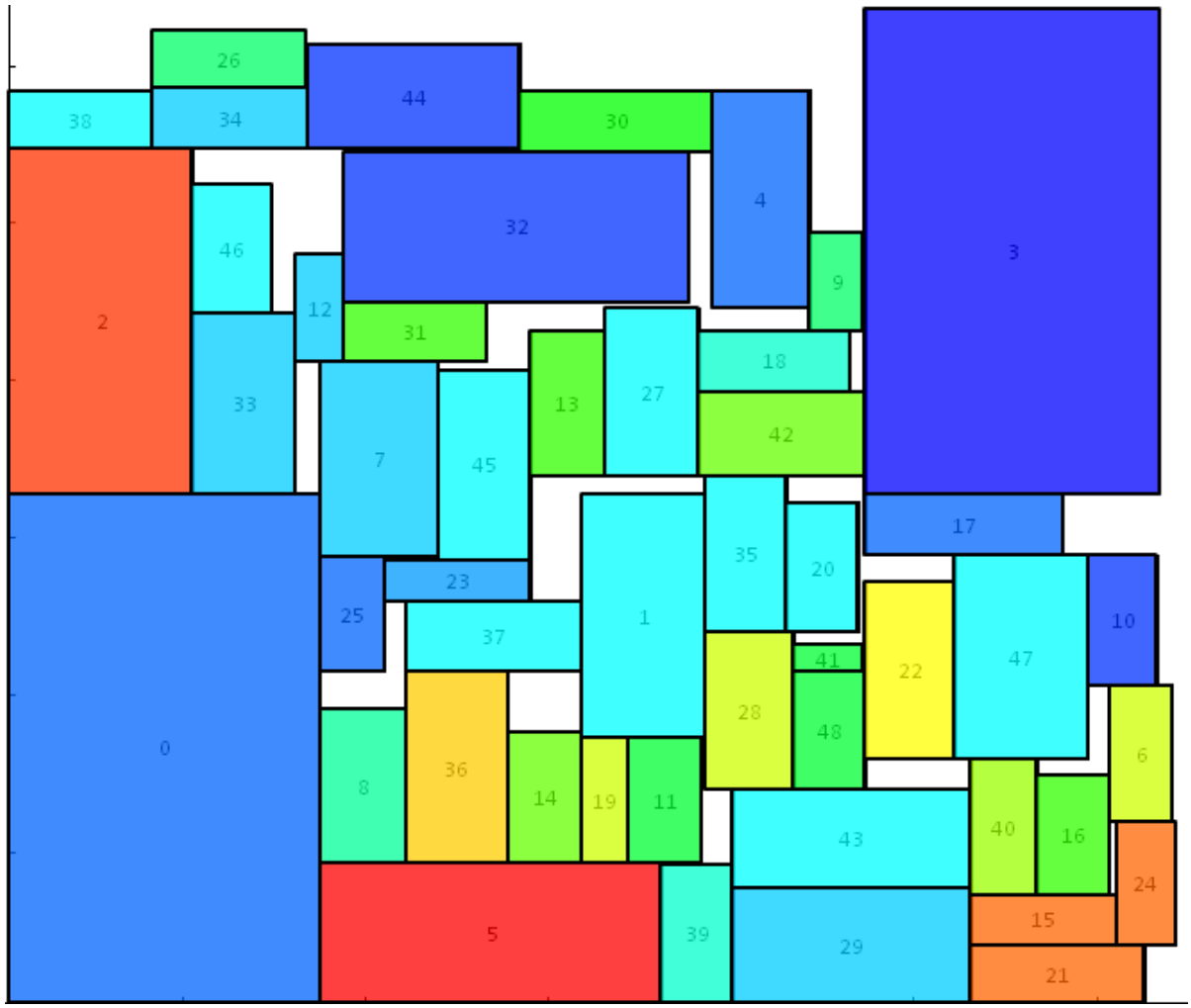


図 4.9: ami49 における従来手法＋アウトライン制約＋温度評価の結果.

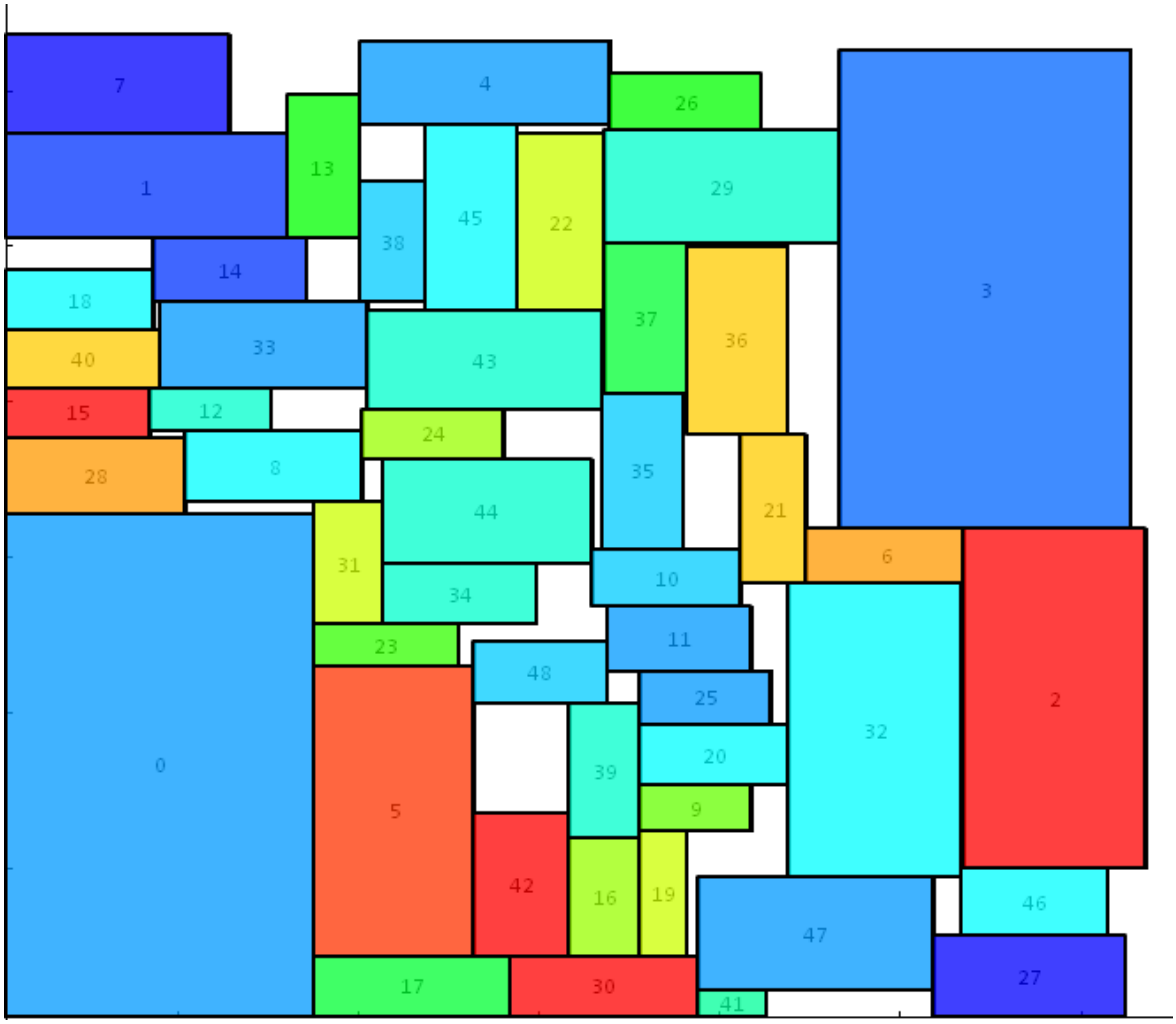


図 4.10: ami49 における提案 GA の結果.

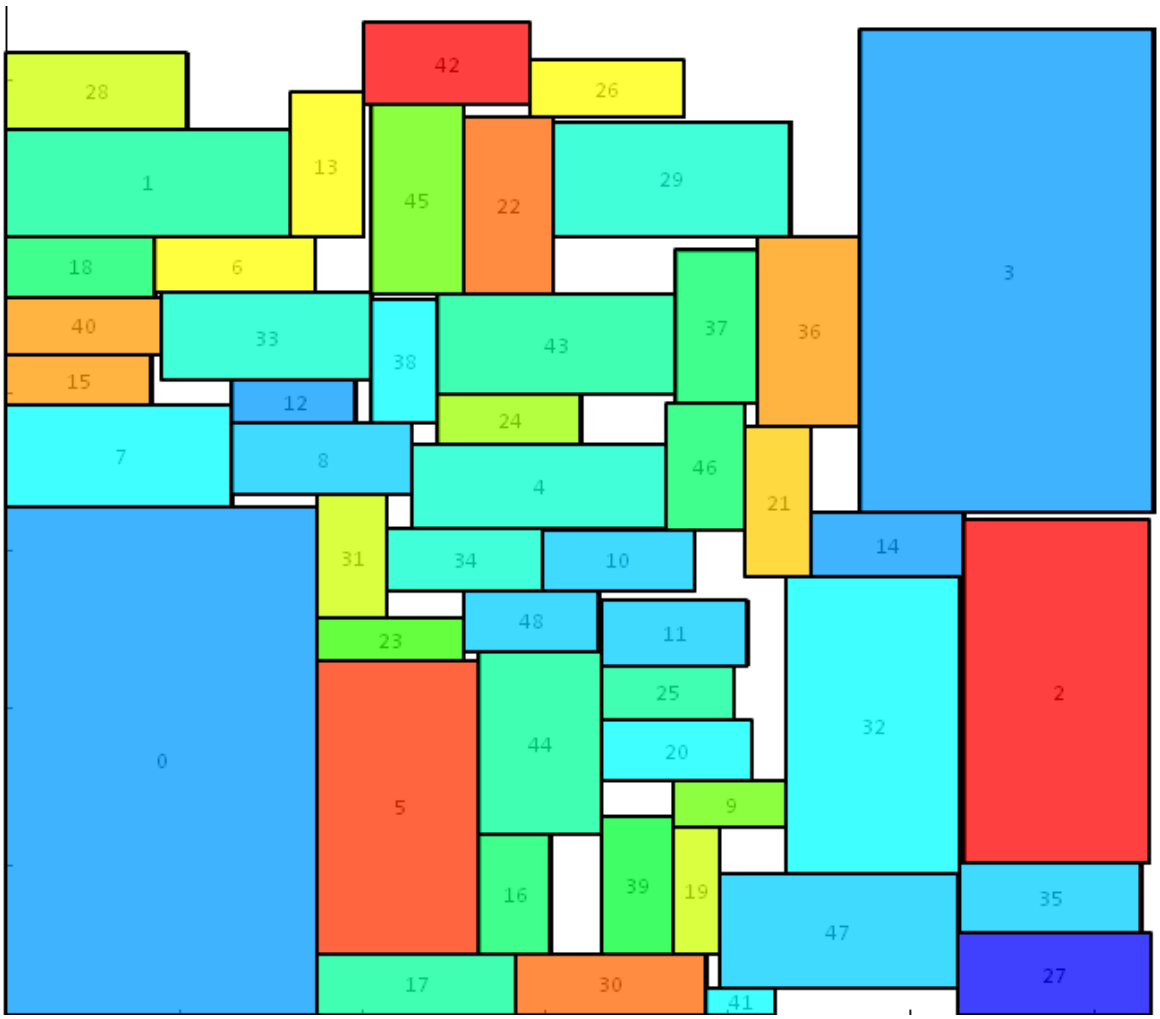


図 4.11: ami49 における置き換え手法の適用結果.

4.4 本章のまとめ

本章では，第 3 章で述べた提案手法を実装し，ベンチマークとして MCNC ベンチの中から ami33，及び ami49 を用いてシミュレーションを行った．実験ではまず初めに，エリート度に基づく遺伝的アルゴリズム，この GA に温度を考慮したコスト関数を適応したもの，提案 GA の 3 つについて比較を行った．その結果，動作温度計算と評価関数，そして局所改良の 3 つを改良した提案 GA では，ami33 では最大で 4.0 ℃，ami49 では最大で 4.9 ℃の低温化という結果を得る事が出来た．

次に，提案 GA の結果に対してモジュール置き換えを行う様子を図で説明し，更にその結果を示した．提案手法（提案 GA ＋モジュール置き換え）では，一度温度評価を行って実際の温度分布を見てから置き換えを行うため効率良く温度改善を図る事が出来た．この提案手法を用いた事により，ami33 では最大 4.7 ℃，ami49 では最大で 5.8 ℃の低温化を確認する事が出来た．

第5章

結論

本論文では，発熱量の低減を目的としたフロアプラニング手法として，エリート度に基づく遺伝的アルゴリズムを用いた．そして熱を考慮した評価関数の設定手法と局所改良法を新たに提案し，前述の GA に適用した．また，GA で求めたフロアプランを対象としたモジュール置き換え手法の提案も行った．そしてこれらの手法に対し計算機実験を行い，その結果を示した．

第 2 章「**エリート度に基づく遺伝的アルゴリズム**」では，本手法のベースであるエリート度に基づく遺伝的アルゴリズムについて説明した．この手法ではデータ構造としてシーケンスペアを用いており，これに対して選択，交叉，突然変異の 3 ステップを行っている．交叉方法はエリート交叉（CPTX）と非エリート交叉（PPEX）の 2 種類があり，優秀な局所配置を受け継ぐエリート個体に対してはそれを保存するために CPTX を行い，非エリートの個体に対しては PPEX を行う．PPEX では，両親の個体それぞれに対して任意モジュールを中心とする窓領域を生成し，領域内のモジュール順を相手親の順序に変更する．

第 3 章「**提案フロアプラニング手法**」では，動作温度の計算方法及び，GA における評価関数の設定手法と局所改良法を提案した．動作温度の計算では，計算の際に監視すべきモジュールの選出方法について，消費電力密度だけでなく，モジュールの周の長さも考慮する方法を提案した．

また，評価関数の設定では，エリート度に基づく遺伝的アルゴリズムの評価関数に対して改変を行った．2.2.2 節における評価関数では面積と総配線長の 2 つのみを考慮していたが，本章では発熱量の低減という目的に沿った，より洗練されたフロアプランを生成するために，既存の評価関数の中から温度に関係しない要素を出来るだけ取り除き，温度に関係する要素を加えた新しい評価関数を提案した．具体的には，まず既存の評価関数から面積を除きこれをアウトライン制約として与えた．次に 2 つ目の要素として 3.2.1 節で計算した温度を，3 つ目としてホットモジュールに隣接するモジュール数を評価関数に加えた．このように設定を行い出来るだけ温度にフォーカスする事で，より本研究の目的にあったフロアプランが生成される事が見込まれる．

また局所改良法では，面積制約を満たす個体に対してホットモジュールに隣接するモジュールの回転を行い，評価値の改善を行った．

そして，GA により生成されたフロアプランに対して，ホットモジュールに隣接する中で最も温度の高いモジュールを他のモジュールと置き換える事で発熱量を抑えるモジュール置き換え手法を提案した．

第 4 章「**計算機実験**」では，第 3 章で述べた提案手法を実装し，ベンチマークとして MCNC ベンチの中から ami33，及び ami49 を用いてシミュレーションを行った．実験では，エリート度に基づく遺伝的アルゴリズムとこの GA に温度を考慮したコスト関数を適応したもの，提案 GA，そして提案手法（提案 GA ＋モジュール置き換え）の 4 つの比較を行った．その

結果，提案手法を用いた事により，ami33 では最大 4.7 ℃，ami49 では最大 5.8 ℃の低温化を確認する事が出来た．

提案手法の問題点と今後の課題について述べる．問題の 1 つ目は配線長の増加である．提案手法では従来のエリート度に基づく遺伝的アルゴリズムに比べて大きく配線長が増加している．これはコスト関数における重みを変化させるか，制約として予め上限値を設定しておく必要があると考えられる．

問題の 2 つ目は提案手法が GA とその後のモジュール置き換えという 2 ステップに分かれている事である．本来ならば GA における探索で最適解が見つかる筈であるため，モジュール置き換えによって値が改善するという事は GA での探索が不十分であるという事になってしまう．そのため，今回の提案では触れなかったが，交叉や突然変異のステップにも変更を加える必要があると考えられる．

問題の 3 つ目はマルチコアプロセッサを考慮していない点である．現在ではマルチコアが主流であるため，複数コアを 1 パッケージに纏める際の配置も考える必要がある．

謝辞

本論文全般にわたり，御指導ならびに御助言を授かった大附辰夫教授，柳澤政生教授，戸川望教授に深く感謝いたします.

最後に，本論文に関する研究活動全般にわたり支援していただいた大附研究室，柳澤研究室および戸川研究室の皆様に感謝いたします.

参考文献

[1] S. Adya and I. Markov, "Fixed-outline floorplanning: enabling hierarchical design", in *IEEE Transactions on Very Large Scale Integration(VLSI) systems*, 2003.

[2] D. Chatterjee , W. Manilas and I. Markov, "Cooler- a fast multiobjective fixed-outline thermal floorplanner", in *Proc of Austin Conference on Integrated Systems and Circuits*, 2008.

[3] Y. Chen and Y. Li, "Temperature-aware floorplanning via geometric programming", in *Mathematical and Computer Modelling*, 2010.

[4] T. Chen and Y. Chang, "Packing Floorplan Representations", http://74.125.155.132/scholar?q=cache:OPVNLD5Nr3QJ:scholar.google.com/+Packing+Floorplan+Representations&hl=ja&as_sdt=2000, 2008.

[5] D. Chen, C. Lin, Y. Wang and C. Cheng, "Fixed-outline floorplanning using robust evolutionary search", in *Engineering Applications of Artificial Intelligence*, 2007.

[6] J. Choi, C. Cher, H. Franke, H. Hamann, A. Weger and P. Bose, "Thermal-aware task scheduling at the system software level, "in *Proc of the 2007 international symposium on Low power electronics and design*, 2007.

[7] 舟堀 浩介, 檀 良, "選択的突然変異を用いた遺伝的アルゴリズムによるフロアプラン設計問題", 電子情報通信学会技術研究報告, 2000.

[8] 清田 紘司, 藤吉 邦洋, "Sequence-pair 表記された一般構造フロアプランの simulated-annealing 法探索", 電子情報通信学会論文誌, 2001.

[9] Y. Han and Y. Kolen, "Simulated annealing based temperature aware floorplanning", in *J.Low Power Electronics*, 2007.

[10] Y. Han, I. Koren and C. Moritz, "Temperature aware floorplanning", in *Proc of Second Workshop on Temperature-Aware Computer System*, 2005.

[11] M. Healy, M Vittes, M. Ekpanyapong, C. Ballapuram, S. Lim, H. Hsin, S. Lee and G. Loh, "Microarchitectural floorplanning under performance and thermal tradeoff", in *Proc of the Asia and South Pacific Design Automation Conference* , 2007.

[12] W. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T. Theocharides, and M. Irwin, “ Thermal-aware floorplanning using genetic algorithms ”, in *Sixth International Symposium on Quality of Electronic Design (ISQED ’05)*, 2005.

[13] W. Hung, Y. Xie, N. Vijaykrishnan, M. Kandemir, and M. Irwin, ”Thermal-aware task allocation and scheduling for embedded systems”, in *IEEE/ACM DATE, Munich,Germany*, 2005.

[14] M. Mitchell, H. Lee, G. Loh and S. Lim, ”An introduction to genetic algorithms”, *First MIT Press paperback edition*, 1988.

[15] 村田 患彦, 界外 志織, 石渕 久生, ”多目的最適化問題のための遺伝的局所探索法における優越関係に基づく解変更ルール”, 情報処理学会論文誌, 2004.

[16] 中矢 伸吾, 若林 真一, 小出 哲士”適応的遺伝的アルゴリズムとシーケンスペアに基づくフロアプラニング手法”, 情報処理学会研究報告, 1999.

[17] K. Skadron, M Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, ”Temperature-aware microarchitecture”, in *Proc of the 30th International Symposium on Computer Architecture*, 2003.

[18] J. Srinivasan, V. Adve, P. Bose, J. Rivers, ”The Impact of Technology Scaling on Lifetime Reliability”, in *Proc of the International Conference on Dependable Systems and Networks*, 2004.

[19] Y. Yang and K. Li, ”Temperature-aware dynamic frequency and voltage scaling for reliability and yield enhancement”, in *Asia and South Pacific Design Automation Conf*, 2009.

[20] J. Yang, X. Zhou, M. Chrobak, and Y. Zhang, ”Dynamic thermal management through task scheduling”, in *IEEE International Symposium on Performance Analysis of Systems and Software*, 2008.